BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 4 : G06F 1/00, H04M 3/00 — A1

(11) International Publication Number: WO 89/12271

(43) International Publication Date: 14 December 1989 (14.12.89)

(21) International Application Number: PCT/US89/02250

(22) International Filing Date: 24 May 1989 (24.05.89)

(30) Priority data:
198,981 26 May 1988 (26.05.88) US

(71) Applicant: GENESIS ELECTRONICS CORPORATION [US/US]; Lake Forest Technical Center, 103 Woodmere Road, Folsom, CA 95630 (US).

(72) Inventors: MILLINGTON, James, E. ; 8331 Lancraft Drive, Sacramento, CA 95823 (US). SAUNDERS, Jonathan, R. ; 650 Flower Drive, Folsom, CA 95630 (US). PFEIFFER, Randall, R. ; 9885 Country Park Court, Roseville, CA 95661 (US). PARKER, Martin, F. ; 8360 W. Hidden Lakes Drive, Roseville, CA 95661 (US). HOOPER, William, G. ; 8927 Central Avenue, Orangevale, CA 95662 (US). O'DONNELL, Charles, A. ; 8301 Mondon Way, Orangevale, CA 95662 (US).

(74) Agents: EQUITZ, Alfred, A. et al.; Limbach, Limbach & Sutton, 2001 Ferry Building, San Francisco, CA 94111 (US).

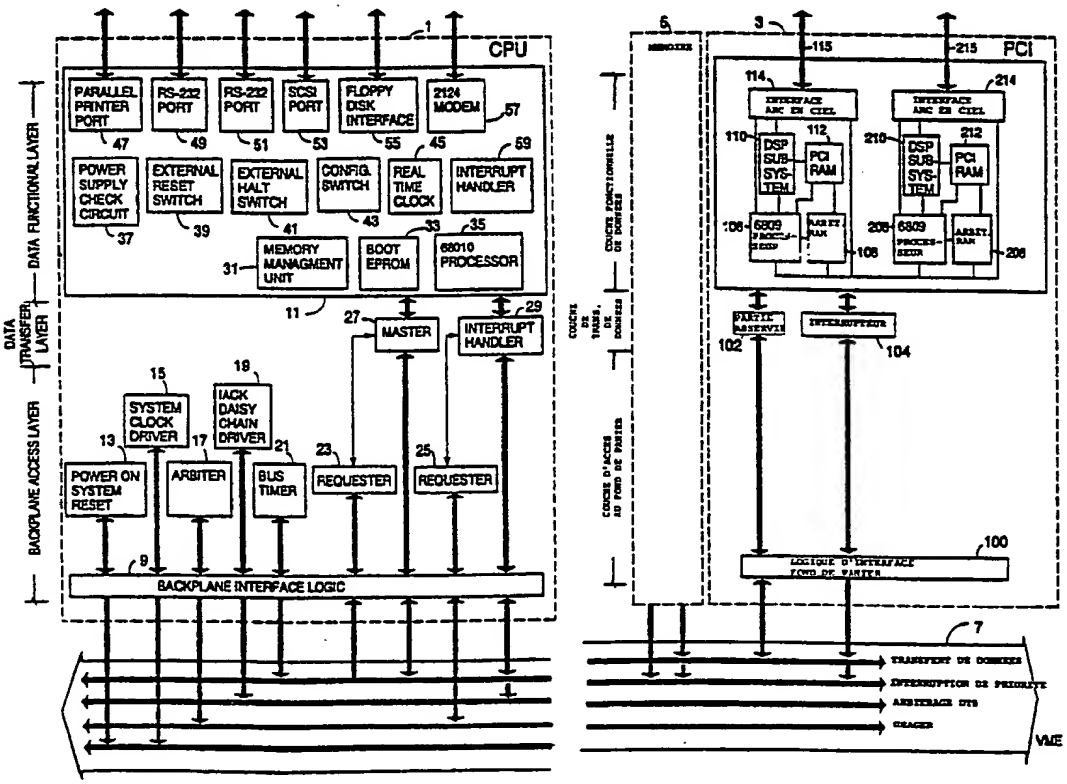
(81) Designated States: AU, BE (European patent), DE (European patent), DK, FR (European patent), GB (European patent), IT (European patent), NL (European patent), NO, SE (European patent).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: VOICE INFORMATION SYSTEM AND METHOD FOR TELEPHONE APPLICATIONS

(57) Abstract

A voice processing system and method for performing a variety of operations, including voice mail processing, in response to digital signals, including digitized voice signals and DTMF signals. A preferred embodiment of the invention includes a CPU (1), a number of programmable communications interface units (3), and a VME bus (7) connecting the CPU with the programmable communication interface (PCI) units. Each PCI unit interfaces with one or more telephone lines. The CPU and the PCI units are controlled by a software operating system in which both digitized voice signals and other digital data may coexist, for efficient processing in a single file structure.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FI	Finland	ML	Mali
AU	Australia	FR	France	MR	Mauritania
BB	Barbados	GA	Gabon	MW	Malawi
BE	Belgium	GB	United Kingdom	NL	Netherlands
BF	Burkina Fasso	HU	Hungary	NO	Norway
BG	Bulgaria	IT	Italy	RO	Romania
BJ	Benin	JP	Japan	SD	Sudan
BR	Brazil	KP	Democratic People's Republic of Korea	SE	Sweden
CF	Central African Republic			SN	Senegal
CG	Congo	KR	Republic of Korea	SU	Soviet Union
CH	Switzerland	LI	Liechtenstein	TD	Chad
CM	Cameroon	LK	Sri Lanka	TG	Togo
DE	Germany, Federal Republic of	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		
ES	Spain	MG	Madagascar		

VOICE INFORMATION SYSTEM AND METHOD FOR TELEPHONE APPLICATIONS

Field of the Invention

The invention relates to voice processing systems for receiving telephoned voice signals, digitizing the voice signals, and processing the digitized signals. More particularly, the invention is a system and method for performing a variety of voice processing operations, including voice mail and automated call forwarding operations, and processing both digitized voice data and other digital data.

Background of the Invention

Conventional voice processing systems include circuitry for digitizing voice signals, circuitry for generating analog audio signals from stored digital signals, and one or more computers for processing both digitized voice signals and high-frequency encoded telephone keypad command signals. Such telephone keypad command signals will be referred to herein as "dual tone multiple frequency" or "DTMF" signals. A voice mail system is one type of conventional voice processing system.

A conventional voice mail system is described in U.S. Patent 4,371,752, issued February 1, 1983 to Matthews, et al., and U.S. Patent 4,652,700, issued March 24, 1987 to Matthews, et al. This voice mail system allows users at remote telephone locations to record voice messages for designated addressees, and to play back and edit the recorded messages by transmitting preassigned DTMF codes over telephone lines. A user may also access the system by telephone,

enter a preassigned DTMF identification code, and receive any messages that have been recorded on the system for the user. Once a message has been recorded on the system, the system is capable of automatically dialing the message addressee to deliver the recorded message. The system digitizes received analog audio signals before it digitally processes them, and is capable of generating analog audio output signals from stored digital signals.

The system of U.S. 4,371,752 (and 4,652,700) includes CPU 70 (a component of call processor subsystem 62A) and CPU 100 (a component of administrative subsystem 60). Each CPU is connected by a multibus to a number of identical "Universal Control Boards" ("UCBs"), including communication port interface 74, disk adapters 76 and 78, and block transfer interface bus unit 80 (for CPU 70), and disk adapters 114 and 116, and block transfer interface bus unit 118 (for CPU 100). Each UCB includes an Intel 8085 microprocessor which is preprogrammed to establish a definite function for the UCB.

Communication port digital data bus 88 connects communication port interface 74 with a number of communication port drivers 92. A CODEC 96 is connected between each driver 92 and a user telephone 18. Communication port interface 74 accepts digitized audio signals from each CODEC and supplies them on the multibus to the other system components for processing. Interface 74 also supplies digital signals received on the multibus from the other system components to the CODECs for conversion into analog audio format for transmission to the appropriate user telephones.

The system disclosed in U.S. 4,371,752, however, has a severely limited range of applications. The system is capable of functioning only as a voice mail

system, with an extremely limited menu of user-selected operations, and with each individual port driver 92 performing a fixed, identical function. The port drivers are not independently programmable to perform programmed functions, and there is no capability for the system users to program a system CPU to process both digitized voice messages and other digital data stored in a common file structure in the system, for example to generate customized output signals, such as voice reports.

Another conventional voice mail system is disclosed in U.S. Patent 4,625,081, issued November 25, 1986 to Lotito, et al. The system of U.S. 4,625,081 includes an information processing system 250 which in turn includes one or more general purpose processors (942 and 944) said to be capable of executing application programs. The system also includes a number of real-time subsystems (230 through 238) for receiving voice and control signals from telephone room subsystems 206, 214, and 216, providing switching connections between channels, and communicating with processing system 250 to facilitate storage and retrieval of voice messages. However, the disclosed applications of the system are limited to use as a voice mail system with an extremely limited menu of user-selected operations. There is no suggestion that the system be used to process both digitized voice messages and other digital data stored in common file structure in the system, for example to generate customized output signals, such as voice reports.

Yet another conventional voice mail system is disclosed in U.S. Patent 4,549,047, issued October 22, 1985 to Brian, et al. The U.S. 4,549,047 system includes a number of digital computers 22 each communicating with a telephone signal conversion

subsystem 40 via a digital data bus 24. Subsystem 40 receives Touch-Tone telephone signals and analog signals on telephone lines 94, digitizes the analog voice signals received on lines 94, and forwards the received telephone signals in digital form to one of computers 22. In response to such signals, computer 22 may store or retrieve a digital voice message. However, U.S. 4,549,047 does not disclose applications other than a limited menu of voice mail applications, and in particular, fails to disclose the operation of processing both digitized voice messages and other digital data stored in a common file structure in the system, for example to generate customized output signals, such as voice reports. Further, U.S. 4,549,047 teaches that computer 22 should preferably be a Digital Equipment PDP-11 computer. Such a conventional, general-purpose computer is not specially adapted for streamlined voice processing applications.

Another type of conventional voice processing system is disclosed in U.S. Patent 4,716,583, issued December 29, 1987 to Groner, et al. The system of U.S. 4,716,583 detects incoming telephone calls from users, initiates telephone calls to users, decodes DTMF commands from users, and translates ACSII text into analog voice signals for transmission on telephone lines to users (for example, in response to DTMF commands from the users). The system includes a central CPU 12 which is connected by a data bus to a number of host computers 23 and to a text-to-speech processing unit 25 for each host computer. A telephone line 27 is connected to each processing unit 25. CPU 12 may forward ASCII data from a host computer 23 to one of the processing units 25 for conversion into an analog voice signal. Unit 25 may then forward the analog voice signal to a user over one of the telephone lines 27. DTMF command signals may

flow from each telephone line 27 to CPU 12. In response to the DTMF command signals, CPU may execute an application program, such as an electronic mail program.

The system of U.S. 4,716,583, however, is not capable of accepting analog voice messages on telephone lines 27, digitizing such voice messages, and storing them (for later transmission in digital form to one of the host computers or reconversion to analog form for retransmission in analog form on phone lines 27). Nor is the system of U.S. 4,716,583 capable of processing digitized telephone voice messages together with other digital data. Furthermore, the computer programs employed by each CPU 31 in the U.S. 4,716,583 system reside in a read-only-memory 35, so that each text-to-speech processing unit 25 has a preassigned function which may not be changed, for example, by downloading new software from central CPU 12.

It has not been known until the present invention how to design a voice processing system which avoids the limitations described above, and which is capable of functioning not only as a voice mail system, but also as a general purpose data processing system for processing both digitized voice signals and other digital data (to generate, for example, customized voice reports), and storing both digitized voice signals and other digital data together in a single record.

Summary of the Invention

The invention is voice processing system and method for performing a variety of operations, including voice mail processing, in response to digital signals, including digitized voice signals and DTMF telephone keypad signals. A preferred embodiment of the inventive system includes a CPU, a number of

programmable communications interface ("PCI") units, and a bus connecting the CPU with the PCI units. Each PCI unit interfaces with one or more telephone lines, and includes means for digitizing analog signals received on each telephone line and for converting digital signals from the bus into analog form for transmission on selected ones of the telephone lines.

The CPU is specially designed to communicate extremely rapidly with devices coupled with the CPU on an SCSI bus.

The CPU and the PCI units are controlled by a software operating system in which both digitized voice signals and other digital data may coexist, for efficient processing in a single file structure. The operating system employs novel data constructs known as "pseudo-files" and files known as "tagged messages". One important type of tagged message is a file which includes not only a voice message, but also instructions to the operating system for causing playback of voice prompts inviting the voice message recipient to send either a voice mail reply or to execute a customized application program. By using pseudo-files, the inventive system facilitates rapid file creation, which advantageously minimizes the perceptible delay to a system user desiring to record a voice message as an operating system file.

The operating system of the invention also allows a user to customize a voice menu of options to be vocally presented to persons calling one of the PCI units. Examples of such options include execution of a voice mail application program, or another, custom designed application program, by sending DTMF and vocal signals over a telephone line.

The function of each PCI may be independently defined by software downloaded from the CPU. The

invention comprises improved methods, preferably implemented in software, for automated monitoring and transferring of incoming telephone calls.

Brief Description of the Drawings

Figure 1 is a block diagram of a preferred embodiment of the inventive system.

Figure 2 is a schematic diagram of the VME slave interface and interrupter of PCI unit 3 of the Fig. 1 embodiment.

Figure 3 is a schematic diagram of the random access memory components of PCI unit 3 of the Fig. 1 embodiment.

Figure 4 is a schematic diagram of the port processor and digital signal processor components of PCI unit 3 of the Fig. 1 embodiment.

Figure 5 is a schematic diagram of the CODEC, DTMF transceiver, and telephone line interface components of PCI unit 3 of the Fig. 1 embodiment.

Figure 6 is a schematic diagram of a first portion of CPU 1 of the Fig. 1 embodiment.

Figure 7 is a schematic diagram of the remaining portion of CPU 1 of the Fig. 1 embodiment.

Figure 8 is a flow chart of a preferred embodiment of the manner in which the inventive system processes an incoming telephone call.

Figure 9 is a flow chart of a preferred embodiment of an inventive method for automated monitoring of the progress of a telephone call.

Figure 10 is a flow chart of a preferred embodiment of a portion of the Figure 9 method.

Figure 11 is a flow chart of a preferred embodiment of a portion of the Figure 10 method.

Figure 12 is a flow chart of a preferred embodiment of a portion of the Figure 11 method.

Figure 13 is a flow chart of a preferred embodiment of a portion of the Figure 9 method.

Figure 14 is a flow chart of a preferred embodiment of an inventive method for automated call transfering.

Detailed Description of the Preferred Embodiments

Figure 1 is a block diagram of a preferred embodiment of the invention. Central processing unit ("CPU") 1, programmable communications interface ("PCI") unit 3, and memory unit 5 each interface with bus 7. In a preferred embodiment, bus 7 is a conventional VME bus and memory unit 5 is selected from those commercially available which are campatible with the selected embodiment of bus 7. Although one PCI unit 3 is shown, the inventive system may include more than one identical PCI units, each connected in the same manner to bus 7 as is unit 3 in Figure 1.

Each PCI unit 3 includes an interface circuit 114 for receiving signals from, and transmitting signals to, telephone line 115. Interface circuit 114 includes a CODEC for digitizing incoming voice signals, a telephone line interface, and a DTMF detector for receiving incoming DTMF signals.

Each PCI unit 3 also includes a digital signal processor 110, which performs speech compression and reconstruction; a port processor 106 (preferably a 2MHz Motorola 6809 integrated circuit), which performs all port processing operations other than speech compression and reconstruction, including data transfers, telephone line interface operations, and call progress detection operations; a random access memory ("RAM") unit 112; a RAM arbiter 108; a VME bus slave interface unit 102; a VME bus interrupter unit 104; and a backplane interface

logic unit 100 for interfacing between VME bus 7 and units 102 and 104.

In the preferred embodiment shown in Figure 1, each PCI unit 3 also includes a second interface circuit 214 for transferring signals to and from telephone line 215, a second port processor 206, a second digital signal processor 210, an second RAM unit 212, and a second RAM arbiter 208. Each of units 206, 208, 210, 212, and 214 is identical to the corresponding one of units 106, 108, 110, 112, and 114, and units 206, 208, 210, 212, and 214 are connected to each other and to units 102 and 104 in the same manner as are corresponding units 106, 108, 110, 112, and 114.

CPU 1 performs all system processing for the inventive voice processing system. In a preferred embodiment, CPU 1 includes a microprocessor 35 (which is preferably a Motorola MC68010 integrated circuit), a memory management unit 31 (which is preferably a Motorola MC68451 integrated circuit), and logic for supporting system input/output and VME bus supervisory functions. The circuitry supporting the VME bus supervisory functions includes system clock driver 15, bus arbiter 17, IACK daisy chain driver 19, and bus timer 21.

Each PCI unit 3 must be compatible with bus 7. In a preferred embodiment in which bus 7 is a conventional VME bus, unit 102 is a VME bus slave interface unit which appears to bus 7 as a slave with 16 bit and 8 bit data transfer capability, block transfer capability, and 24 bit addressing capability. A preferred embodiment of unit 102 will be discussed below with reference to Figure 2.

Port processor 106 (or port processor 206) of PCI unit 3 may interrupt system processor 35 of CPU 1 via VME bus interrupter unit 104. A preferred embodiment of

unit 104 will be described below with reference to Figure 2.

Interface logic unit 100 comprises conventional VME backplane interface circuitry for connecting units 102 and 104 with VME bus 7.

Figure 2 is a schematic diagram of a preferred embodiment of VME bus slave interface unit 102 (comprising programmable logic devices U16, U21, and U35, and integrated circuit U73), VME bus interrupter unit 104 (comprising programmable logic devices U42 and U49, and registers U3 and U4), and interface logic unit 100 (comprising buffer circuits U17, U28, U29, U6, U11, U43, U50, and U78). The circuits comprising interface logic unit 100 are connected between VME bus 7 and units 102 and 104.

Programmable logic devices U21 and U35 of slave unit 102 decode the buffered VME bus address signals (VA01-VA23) and address modifier signals (AM0-AM5). Device U35 compares signals VA01-VA23 with the slot identification signals SID0-SID3 received from VME bus 7 for identifying the slot in which the PCI unit 3 is installed (typically, a number of PCI units 3 will be installed in different slots on a motherboard connecting all components of the system of the invention). If the address signals match the slot identification number, the VALIDID line goes high. Device 35 also decodes the address modifiers AM0-AM5, and causes line VALIDAM* to go low, for example, if any of the address modifiers indicates a standard non-privileged data, program, or block transfer access, or a supervisory data, program, or block transfer access.

Device U21 decodes address signals VA21-23 and VA14-16. If the address falls within the PCI address space, and VIACK is low and VALIDID is high, the PCI is selected. If the PCI is selected with an invalid

address modifier (i.e., VALIDID is low), BADXFER* will go low, indicating that an illegal transfer is being attempted. This will cause the "DTACK and BERR generator" (circuit U16) to generate a VME bus error signal. Device U21 also generates the following signals: AVRAMSEL* (for selecting RAM unit 112), BVRAMSEL* (for selecting RAM unit 212), CLATCHSEL* (which is active when the VME bus master, CPU 1, selects port control latch circuit U79), and LATCHRD* (the VME interrupt vector register select signal, which is active when the VME bus master addresses the VME bus interrupt vector register of circuit 104).

Table 1 specifies the logical relationships between the input and output signals of device U21, and Table 2 specifies the logical relationships between the input and output signals of device U35. In Tables 1 and 2 (and the other Tables set forth below), the symbol "!" represents "inverse" (so that signal !A is the inverse of signal A), the symbol "&" represents "and", and the symbol "#" represents "or".

Table 1

```
!AVRAMSEL =    VA23 & VA22 # !VA21 & VALIDID & !VA16 &
               VA15 & VA14 & !VIACK & VLWORD & VAS &
               !VALIDAM;
!BVRAMSEL =    VA23 & VA22 & !VA21 & VALIDID & VA16 &
               VA15 & VA14 & !VIACK & VLWORD & VAS &
               !VALIDAM;
!BADXFER  =    (VA23 & VA22 & !VA21 & VALIDID & VAS &
               !VIACK & !VLWORD # VA23 & VA22 & !VA21 &
               VALIDID & VAS & !VIACK & VALIDAM);
!LATCHSEL =    VA23 & VA22 & !VA21 & VALIDID & !VA15 &
               !VA14 & & !VIACK & VLWORD & VAS &
               !VALIDAM & VDS0 & VWRITE;
!VRAMSEL  =    VA23 & VA22 & !VA21 & VALIDID & VA15 &
               VA14 & !VIACK & VLWORD & VAS & !VALIDAM;
```

```
!LATCHRD  =   VA23 & VA22 & !VA21 & VALIDID & !VA15 &
              VA14 & !VIACK & VLWORD & VAS & !VALIDAM &
              VDS0 & !VWRITE;
```

Table 2

```
!VALIDID  =   ((((((VA20 & !SID3
              # !VA20 & SID3)
              # VA19 & !SID2)
              # !VA19 & SID2)
              # VA18 & !SID1)
              # !VA18 & SID1)
              # VA17 & !SID0)
              # !VA17 & SID0);
VALIDAM   =   !(AM0 & AM3 & AM4 & AM5
              # AM1 & AM3 & AM4 & AM5)
```

VME bus 7 is asynchronous, so that all transfers require a handshaking between the VME bus master (CPU 1) and each VME bus slave (each of PCI units 3). The handshaking signals employed in a data transfer are: AS* (address strobe), DS0* and DS1* (data strobe signals), DTACK* (a data transfer acknowledgement signal, the inverse of DTACK), and BERR* (a bus error signal).

Device U16 of PCI unit 3 generates the following signals: DTACK (which occurs when VDTACK is active, or when the control latch is selected, or during an interrupt acknowledge, and remains active until both VDS0 and VDS1 are inactive); BERR (the inverse of BERR*, which has the same timing as DTACK, and which becomes active when an illegal access is attempted, as described above); LATCHCK and DLCLK (state variables of a simple state machine. DLCLK goes low for one clock cycle when the control latch is accessed, and LATCHCK goes low during the next clock cycle); VDBEN (an asynchronous signal which goes active whenever any

component of the PCI is accessed and remains low until both VDS0 and VDS1 are high); and VALD and VACLK (state variable signals of an asynchronous state machine. VACLK clocks the VME bus address into the address counters when VALD is low, or increments the counters when VALD is high).

Table 3 is the state transition table for signals VALD and VACLK:

Table 3

Outputs (VALD,VACLK) \ Inputs (VAS,VDS0,DTACK)	000	001	011	010	110	111	101	100
00	00*	00*	00*	00*	01	01	01	01
01	00	00	00	00	11	11	11	11
11	00	00	00	00	10	11*	10	10
10	00	00	00	00	10*	11	10*	10*

* = Stable state

Each entry in Table 3 represents the next state of the asynchronous state machine given the current inputs and the current outputs. A state is "stable" if the next state is identical to that state. As is apparent from Table 3, the address counters will load whenever VAS goes active, and will increment on every odd byte access as long as VAS remains active.

Table 4 specifies the logical relationships between the input and output signals of device U16:

Table 4

Original Equations:

```
!VALD   =  (!VAS # !VACLK & !VALD);
!VACLK  =  (((!VAS # VALD & !VDS0) # VALD & !DTACK) #
           VALD & VRAMSEL);
!VDBEN  =  (((((VDS1 & !VRAMSEL
```

```
                # VDS0 & !VRAMSEL)
                # VDS0 & !LATCHSEL)
                # !INTACK)
                # !VDBEN & VDS1)
                # !VDBEN & VDS0);
!BERR   =       ((!BERR & BADXFER # BERR & !VDS0 & !VDS1) #
                VRESET);
!DTACK  =       (((!DTACK & VDTACK & INTACK & LATCHCK
                # DTACK & !VDS0 & !VDS1)
                # !DTACK & !INTACK & DLCLK)
                # VRESET);
!LATCHCK =      !DLCLK & !LATCHSEL & !VRESET;
!DLCLK  =       (VDS0 & !LATCHSEL & DLCLK & LATCHCK & !DTACK &
                !VRESET # DLCLK & !INTACK & !DTACK & !VRESET);
```

System processor 35 of CPU may control operation of PCI 3 via port control latch U79. Device U79 is preferably a Texas Instruments Inc. 74LS259B addressable latch. When signal CLATCHSEL*, emerging from device U79 goes low, signal VD00 is latched at the output addressed by the three select input signals VA16, VA01, and VA02 (of which VA16 is the most significant select input). VA16 separates the eight possible inputs into two groups of four, addressed 64K bytes apart. Each of these two groups controls one of ports A and B. The control outputs of device U79 are: AOFFH (BOFFH), which is supplied to device U89 of the port A interface 114 shown in Figure 5 (or the corresponding circuit of port B interface 214), for forcing the signalling relays of interface 114 (214) to an "off hook" condition when high, provided that signal A6809 (B6809) is low; A6809 (B6809), which gives port processor 106 (206) control over the signalling relays of interface 114 (214) when high; AMRST (BMRST), which resets port processor 106 (206) when low; and AMINT (BMINT), which generates an interrupt to port processor 106 (206) when pulsed.

When the system powers up, all control bits are cleared, thus resetting port processors 106 and 206 and forcing the signalling relays of interfaces 114 and 214 to their "power up" state, corresponding to a "zero" value of signal AOFFH and BOFFH.

PCI unit 3 may interrupt system processor 35 of CPU 1, by sending interrupt request signals from interrupt unit 104 on VME bus 7. Unit 104 comprises programmable logic devices U42 and U49 (for timing and control), and registers U3 and U4 (for providing the interrupt vectors). Registers U3 and U4 are preferably Texas Instruments Inc. 74LS374 integrated circuits. Interrupt unit 104 is a "release on acknowledge" (ROAK) interrupter in the sense that it will release its interrupt request only when the interrupt has been acknowledged. More specifically, the functions of unit 104 in each PCI unit 3 include: generating an interrupt request on line IRQ, providing an interrupt vector on lines D00-D07 when the interrupt request is being acknowledged, releasing its interrupt request during the acknowledge cycle, passing the IACKIN signal (the incoming interrupt acknowledge daisy chain signal) to the next PCI unit 3 (typically mounted in the next slot on a motherboard) if no interrupt is pending, or if the interrupt being acknowledged is at a different level, and providing the DTACK* signal to end the interrupt acknowledge cycle. In the Figure 2 embodiment, interrupt unit 104 will treat ports A and B (hence port processors 106 and 206) as separate "level 3" interrupters, with port A having priority.

Devices U42 and U49 implement timing and control for interrupter 104. Device U49 generates the following four combinatorial outputs from the input signals received from port processors 106 and 206: LOADA (LOADB), whose rising edge will load data from the port

A (port B) data bus into interrupt vector register U4 (U3); and SETA* (SETB*), which, if low, will set an interrupt request for port A (port B). Device U49 also generates four registered signals, from two state machines, one for each port, according to the state transition table set forth as Table 5:

Table 5

CURRENT STATE					NEXT STATE	
VRESET	LATCHRD	EN*	Q2	RST*	Q2'	RST*'
1	X	X	X	X	1	0
0	X	1	1	1	1	1
0	0	X	1	1	1	1
0	1	0	1	1	0	1
0	1	0	0	1	0	1
0	X	1	0	1	0	0
0	X	1	0	0	1	0
0	X	1	1	0	1	1

Signals Q2A and Q2B (represented by terminals 18 and 20 of device 49, respectively, but used internally only within device 49) will go low during an interrupt acknowledge cycle, but not when system processor 35 is reading the contents of the interrupt register (U3 or U4) during testing. At that time, RSTA* (RSTB*) will go low for two clock cycles, resetting the interrupt request. This ensures that the interrupt request remains valid for the entire interrupt acknowledge cycle, but that an interrupt request is not reset by reading the interrupt vector register under software control.

Table 6 specifies the logical relationships between the input and output signals of device U49:

Table 6

```
Original Equations:
!LOADA =  !AMIVSEL & !AMRW & !AMA0 & E;
!LOADB =  !BMIVSEL & !BMRW & !BMA0 & E;
!SETA  =  !AMIVSEL & !AMRW & AMA0 & E;
!SETB  =  !BMIVSEL & !BMRW & BMA0 & E;
!RSTA  =  (!Q2A & ENA # VRESET);
!RSTB  =  (!Q2B & ENB # VRESET);
!Q2A   =  (RSTA & !ENA & LATCHRD & !VRESET # !Q2A & RSTA
          & !VRESET);
!Q2B   =  (RSTB & !ENB & LATCHRD & !VRESET # !Q2B & RSTB
          & !VRESET);
```

Device U42 generates the following signals: INTACK*, which goes low when either of interrupt vector registers U3 and U4 is enabled, which will occur during an interrupt acknowledge cycle or when the registers are read by system processor 35 for testing purposes; IRQ, which is inverted and then supplied as the interrupt request to the VME bus 7 (signal IRQ is high whenever an interrupt request is pending from either port A or port B); IACKOUT*, the VME bus interrupt acknowledge signal which is output to the next slot in the IACK daisy chain (signal IACKOUT* will go low if VIACKIN is high, and either the current interrupt acknowledge cycle is for an interrupt level other than three, or there is no interrupt request from either port, which latter condition is indicated by DNOIRQ* going low); ENA* and ENB*, which enable their respective interrupt registers, U4 and U3, onto the internal data bus within PCI 3; IRQA* and IRQB*, which are used internally within device U42; DNOIRQ*, which is used internally within device U42 to prevent interrupt circuit 104 from attempting to respond to an interrupt acknowledge cycle if an interrupt request occurs after the cycle has begun (if IRQA* and IRQB* are both inactive when VAS goes active,

DNOIRQ will go low, and DNOIRQ will remain low until VAS goes inactive); and Y1 and Y2, which are state variables of an asynchronous state machine, and are used internally within device U42 (Y2 is employed to qualify several signals, ensuring that they will be delayed for at least one half clock cycle to ensure that IACKOUT* will be delayed until the onboard logic can determine if an onboard interrupt request is pending).

The state transition table for signals Y1 and Y2 is shown below as Table 7:

Table 7

Y1, Y2 \ CLK, VIACKIN	00	01	11	10
11	11*	11*	01	11*
01	11	00	01*	11
00	11	00*	00*	11
10	11	10*	00	11

* = Stable state

Table 8 specifies the logical relationships between the input and output signals of device U42:

Table 8

```
!INTACK   =   (!ENA # !ENB);
!IRQ      =   IRQA & IRQB;
!IACKOUT  =   ((!Y2 & !DNOIRQ & VAS # !Y2 & VAS & !VA2)
              # !Y2 & VAS & !VA1);
!ENA      =   ((!Y2 & !IRQA & VDS0 & VAS & !VA3 & VA2 &
              VA1 & ENB & DNOIRQ
              # !LATCHRD & VDS0 & VA1 & ENB)
              # !ENA & VDS0 & ENB);
!ENB      =   ((!Y2 & !IRQB & IRQA & VAS & !VA3 & VA2 &
              VA1 & VDS0 & ENA & DNOIRQ
              # !LATCHRD & VDS0 & VA1 & ENA)
              # !ENB & VDS0 & ENA);
```

```
IRQA      =    (SETA & IRQA # !RSTA);
IRQB      =    (SETB & IRQB # !RSTB);
!DNOIRQ   =    ((VAS & !DNOIRQ # IRQA & IRQB) # VAS &
               VA3  & !Y2 & VDS0);
Y1        =    (!CLK & Y1 # !VIACKIN);
Y2        =    ((Y1 & Y2 # CLK & Y2) # !VIACKIN);
```

RAM unit 112 may be addressed by either port processor 106 or by CPU 1, with arbitration transparent to both. Similarly, RAM unit 212 may be addressed by either port processor 206 or by CPU 1, with arbitration transparent to both.

Figure 3 is a schematic diagram of the elements shown as blocks 108, 112, 208, and 212 in Figure 1. Each of RAM units 112 and 212 is a shared RAM unit in the sense that both CPU 1 (via VME bus 7) and one of the port processors (unit 106 or 206) may access each of RAM units 112 and 112. RAM unit 112 includes 16K bytes of shared RAM (provided by 8K x 8 static RAM unit U61 and 8K x 8 static RAM unit U63). Similarly, RAM unit 212 includes 16K bytes of shared RAM (provided by 8K x 8 static RAM unit U33 and 8K x 8 static RAM unit U47).

The VME bus address and the port processor address for RAM unit 112 are multiplexed by data selector/multiplexer circuits U62, U68, U74, and U55, each of which is preferably a Texas Instruments Inc. 74LS157 integrated circuit. The VME bus address and the port processor address for RAM unit 212 are multiplexed by data selector/multiplexer circuits U34, U41, U48, and U55, each of which is preferably a Texas Instruments Inc. 74LS157 integrated circuit.

The VME bus 7 interfaces with the "Port A" RAM unit (comprising devices U61 and U73) via bidirectional octal buffers U9 and U15 and octal registers U5 and U10, and with the "Port B" RAM (comprising devices U33 and U47) via bidirectional octal buffers U8 and U14 and octal

registers U5 and U10. Control signals for the interfaces are supplied by the timing and control logic described below.

Port processor 106 interfaces with "Port A" RAM unit 112 (comprising devices U61 and U73) via bidirectional octal buffers U20 and U27, and port processor 206 interfaces with "Port B" RAM 212 (comprising devices U33 and U47) via bidirectional octal buffers U19 and U26. All accesses by the port processors are byte wide synchronous accesses, to be described below.

Buffers U8, U9, U14, U15, U19, U20, U26, and U27 are preferably Texas Instruments Inc. 74LS245 integrated circuits, and registers U5 and U10 are preferably Texas Instruments Inc. 74LS374 integrated circuits.

Programmable logic devices U36, U56, and U57 generate timing and control signals for accessing the RAMs, and "E" and "Q" clock signals for port processors 106 and 206. Identical devices U56 and U57 serve as memory control line decoders, with device U56 associated with "Port A" RAM 112 and device U57 associated with "Port B" RAM 212. Device U36 implements a state machine which runs off the 16KHZ VME system clock ("VSYSCLK"). The state variables and state machine operation are described in Table 9:

Table 9

	CURRENT STATE										NEXT STATE					
	VRESET	VRAMSEL*	VDS	E*	Q	S0	S1	S2	VDTACK*	E*	Q	S0	S1	S2	VDTACK*
1	1	X	X	X	X	X	X	X	X	1	0	0	1	1	1
2	0	X	X	1	0	0	1	1	1	1	0	1	1	1	1
3	0	X	X	1	0	1	1	1	1	1	1	1	1	1	1
4	0	X	X	1	1	1	1	1	1	1	1	0	1	1	1
5	0	X	X	1	1	0	1	1	1	0	1	0	1	1	1
6	0	X	X	0	1	0	1	1	1	0	1	1	1	1	1
7	0	X	X	0	1	1	1	1	1	0	0	1	1	1	1
8	0	X	X	0	0	1	1	1	1	0	0	0	1	1	1
9	0	1	X	0	0	0	1	1	1	1	0	0	1	1	1
10	0	X	0	0	0	0	1	1	1	1	0	0	1	1	1
11	0	0	1	0	0	0	1	1	1	1	0	0	0	1	1
12	0	X	X	1	0	0	0	1	1	1	0	1	0	0	1
13	0	X	X	1	0	1	0	0	1	1	1	1	0	0	1
14	0	X	X	1	1	1	0	0	1	1	1	1	0	1	1
15	0	X	X	1	1	0	0	1	1	1	1	0	1	1	0
16	0	X	1	X	X	X	X	X	0	X	X	X	X	X	0
17	0	X	0	X	X	X	X	X	0	X	X	X	X	X	1

The state variables have the following functions:

Signal E* is the inverse of the E clock required by port processors 106 and 206, and signal E* is employed by devices U56 and U57 to generate the RAM control

signals and to multiplex the addresses from port processors 106 and 206 and VME bus 7;

Signal Q is the Q clock required by port processors 106 and 206;

Signal S0 is employed (with signals E* and Q) to generate the system clocks in a manner to be discussed below, and is also employed as a clock signal for the CODEC control logic (which comprises programmable logic devices U82 and U83);

Signals S1 and S2 become active during a memory access, and signal S2 is also used by the memory control line decoders U56 and U57 to generate RAM control signals and to clock RAM data into the VME data registers;

Signal VDTACK* becomes active immediately following a RAM access to inform the "DTACK" (data transfer acknowledge) signal generator described above that the data transfer is complete;

Signal VRESET is a buffered VME bus reset signal;

Signal VRAMSEL* (which denotes either signal AVRAMSEL* or signal BVRAMSEL*) is active whenever CPU 1 addresses RAM 112 or 212, and is employed to initiate a load of the address counters and to enable the VME bus data buffers; and

Signal VDS denotes either signal VDS1 (VME data strobe "1") or signal VDS0 (VME data strobe "0").

The operation of state machine U36 will next be more fully described with reference to row numbers of Table 9:

Row 1: when VRESET is high, all state variables are set to the indicated values;

Rows 2 through 8: E*, Q, and S0 form a "divide by eight" Gray code counter, and E* and Q provide the overlapping 2MHz clocks required by port processors 106 and 206;

Rows 9 and 10: state machine U36 checks for a request for a RAM access from VME bus 7. If no request is pending, signals S1 and S2 remain inactive, so that any access by the VME master is deferred until the RAM is unused by the relevant port processor;

Rows 11 through 14: if a VME access is requested, S1 will go low for four cycles and S2 will go low for two cycles;

Row 15: following the RAM access, VDTACK* will go low; and

Rows 16 and 17: VDTACK* will remain low until VDS is inactive.

State machine U36 also generates control signals AREN* and BREN*, which enable data buffers U8, U9, U14, and U15 between RAM units U61, U73, U33, and U47 and the VME bus. Signals AREN* and BREN* will go low during a RAM access for two clock cycles during a read operation (coincident with S2), or three cycles during a write operation.

Table 10 specifies the logical relationships between the input and output signals of device U36:

Table 10

```
!E   =  (!E & S0 & !VRESET # Q & !S0 & !VRESET);
!Q   =  ((!Q & !S0 # !E & S0) # VRESET);
!S0  =  ((E & Q # !E & !Q) # VRESET);
!S1  =  (((((!E & !Q & !S0 & !AVRAMSEL & !VRESET &
          VDS0 & VDTACK
          # !E & !Q & !S0 & !AVRAMSEL & !VRESET & VDS1 &
            VDTACK)
          # !E & !Q & !S0 & !BVRAMSEL & !VRESET & VDS0 &
            VDTACK)
          # !E & !Q & !S0 & !BVRAMSEL & !VRESET & VDS1 &
            VDTACK)
          # !S1 & !S2 & !VRESET)
          # !S1 & !Q & !VRESET);
```

```
!S2    =   !S1 & !Q & !VRESET;
!VDTACK = ((!S1 & S2 & Q & !VRESET
                # !VDTACK & VDS0 & !VRESET)
                # !VDTACK & VDS1 & !VRESET)
!AREN      =    (!S2 & !AVRAMSEL # Q & !S1 & !AVRAMSEL &
                VWRITE);
!BREN      =    (!S2 & !BVRAMSEL # Q & !S1 & !BVRAMSEL &
                VWRITE);
```

Devices U56 and U57 decode the state variables from device U36 to generate all the control signals for access to the "Port A" RAM comprising U61 and U73, and to the "Port B" RAM comprising U33 and U47. These control signals include the following:

ARDIR and BRDIR, which determine the direction of transfer, with a low level indicating a write and a high level indicating a read (Each of these signals is determined either by the write line of the port processor ("AMRW" or "BMRW") or the VME bus write signal "VWRITE"). Which source is used depends on the state of signal E*. When signal E* is low, the port processor has access to the RAM, and AMRW (or BMRW) determines the direction. On alternate cycles, VWRITE determines the direction);

VLEN*, which enables the VME data latches U5 and U10, and is active following a read by the VME bus master;

AENML*, AENMH*, BENML*, and BENMH*, which enable data buffers U20, U27, U19, and U26, respectively, between the RAMs and the port processors, with AENML* (BENML*) enabling the low byte when RAM is selected by the port processor, E* is low, and AMA0 (BMA0) is low, and AENMH* (BENMH*) enabling the high byte when AMA0 (BMA0) is high;

AROE* (BROE*) which is the output enable for the

RAM, and is gated by E*, just as is ARDIR (BRDIR), but has opposite sense; and

ARWRL* and ARWRH* (BRWRL* and BRWRH*), which enable the operation of writing one byte of data to RAM 112 (212). During a port processor write operation (when E* is low, and MRW is low), a two cycle write pulse is generated for either the high byte or low byte, depending on AMA0 (BMA0). During a VME write operation, either the high or the low byte, or both, may be written, depending on the state of VDS1 and VDS0.

Table 11 specifies the logical relationships between the input and output signals of both device U56 and identical device U57:

Table 11

```
!RDIR   =   (!E & !MRW # E & VWRITE);
!VLEN   =   !VDTACK & !VWRITE;
!ENML   =   !E & MA0 & !MRAMSEL;
!ENMH   =   !E & !MA0 & !MRAMSEL;
!RWRL   =   (!E & S0 & !MRAMSEL & MA0 & !MRW # !S2 &
            VDS0 & VWRITE & !VRAMSEL);
!RWRH   =   (!E & S0 & !MRAMSEL & !MA0 & !MRW # !S2 &
            VDS1 & VWRITE & !VRAMSEL);
!ROE    =   (!E & MRW # E & !VWRITE);
```

A RAM access by one of the port processors is straightforward. During the half cycle of E* in which the port processor has access, one port data buffer (U20, U27, U19, or U26) is enabled, and one byte, even or odd, is either written or read, depending on the states of AMA0 (BMA0) and AMRW (BMRW). The port processor is never delayed in accessing the RAM.

The VME bus master (CPU 1) may be delayed for as much as one cycle of E* while accessing one of the RAMs. During a read, the RAM data is enabled onto lines VD00-VD15 and latched by units U5 and U10. After the data is latched, U8 and U14 are disabled, U5 and U10 are

enabled, and VDTACK* goes low, signalling that data is available to the VME bus. During a write, the data is simply written to the RAM in the same manner as during a port processor write. Data is latched during a read operation because the VME bus is asynchronous, so that data must be held until the VME bus master (CPU 1) signals that the data is no longer needed.

Figure 4 is a schematic diagram of port processor 106 and digital signal processor 110 of Figure 1. Because processors 106 and 110 are identical to processors 206 and 210, respectively, and are interconnected with interface 114, RAM unit 112, RAM arbiter 108, slave unit 102, and interrupter unit 104 in the same manner as processors 206 and 210 are interconnected with interface 214, RAM unit 212, RAM arbiter 208, slave unit 102, and interrupter unit 104, no separate diagram of processors 206 and 210 is provided.

Port processor 106 includes processor U87 (which preferably is a 2MHz Motorola 6809 integrated circuit) for performing all port processing operations other than speech compression and reconstruction, including data transfers, telephone line interface operations, and call progress detection operations. Processor has access to the resources described below with reference to devices U72, U77, and U85 of Fig. 4.

The port processor address (AMA4-AMA15) is decoded by programmable logic device U72 (which is preferably a PAL12L10 integrated circuit). Table 12 specifies the logical relationships between the input and output signals of device U72:

Table 12

```
!AMIVSEL   =   AMA15 & !AMA14 & !AMA13 & !AMA12 & !AMA7
               & !AMA6 & !AMA5 & AMA4;
!AMTIRQ    =   AMA15 & !AMA14 & !AMA13 & !AMA12 & E &
               AMA7 & !AMA6 & !AMA5 & !AMA4 & !AMRW;
!AMBUFSEL  =   AMA15 & !AMA14 & !AMA13 & !AMA12 & !AMA7
               & !AMA6 & AMA5 & AMA4;
!AMSWCS    =   AMA15 & !AMA14 & !AMA13 & !AMA12 & !AMA7
               & !AMA6 & AMA5 & !AMA4 & AMRW;
!AMVIACS   =   AMA15 & !AMA14 & !AMA13 & !AMA12 & !AMA7
               & !AMA6 & !AMA5 & !AMA4;
!ADTMFIN   =   AMA15 & !AMA14 & !AMA13 & !AMA12 & !AMA7
               & AMA6 & AMA5 & AMA4 & AMRW;
!AMCODRD   =   AMA15 & !AMA14 & !AMA13 & !AMA12 & !AMA7
               & AMA6 & !AMA5 & !AMA4 & AMRW;
!AMCODWR   =   AMA15 & !AMA14 & !AMA13 & !AMA12 & E &
               !AMA7 & AMA6 & !AMA5 & !AMA4 & !AMRW;
!AMRAMSEL  =   AMA15 & AMA14;
```

"Versatile interface adapter" ("VIA") U77 is preferably a Rockwell International A6522A integrated circuit, which performs timing and interrupt functions as well as I/O for the telephone line interface unit 114 (to be discussed below with reference to Figure 5). VIA U77 also functions as the central component of the interrupt structure of port processor 106.

Input signals AMINT and ATINT to VIA U77 are, respectively, the interrupt signals from digital signal processor 110 and the VME bus master (CPU 1). The active edge of these interrupts may be programmed by device U87. When either of these interrupts occurs, the AMIRQA output signal of VIA U77 becomes active. The CODEC timing (to be discussed below with reference to Fig. 5) also asserts a fast interrupt once every 125 microseconds on input line AFIRQ* to device U87, indicating that the CODEC requires service.

Port processor U87 has access to eight status bits (AMD0-AMD7) via octal port U85, indicating respectively: the port ID number (0= Port A, 1= Port B) allowing the port to identify itself as "Port A" or "Port B", Slot ID numbers 0 through 3 identifying the slot into which the PCI unit 3 is installed, and phone interface switch bits ASW1 through AS3 representing the state of the three telephone interface switches S1-S3 shown in Figure 5 (these bits are compared with the requirements of the software installed in U87).

Digital signal processor 110 includes processor U1 (sometimes referred to as "DSP" U1), which preferably is a TMS32010 integrated circuit. U1 accepts data from port processor U87 via a buffer to be described below.

The DSP address (ATA0, ATA1, ATA7, ATA8, ATA9, ATA10, and ATA11) is decoded by programmable logic device U13 (which preferably is a PAL 16L8 integrated circuit). Table 13 specifies the logical relationships between the input and output signals of device U13:

Table 13

```
!ATBUFSEL =   !ATA11 & !ATA10 & !ATA9 & !ATA8 & !ATA7 &
              ATA1 & !ATA0;
!ATINT    =   (!ATA11 & !ATA10 & !ATA9 & !ATA8 & !ATA7
              & !ATA1 & !ATA0 & !ATWEN # ATNINT);
!ATNINT   =   (!ATA11 & !ATA10 & !ATA9 & !ATA8 & !ATA7
              & !ATA1 & ATA0 & !ATWEN # ATINT);
!ATROMSEL =   !ATA11 & !ATA10 & !ATA9 & !ATA8 & !ATA7 &
              !ATMEN;
!ATRAMSEL =   ((((ATA11 # ATA10) #ATA9) #ATA8) #ATA7);
enable NC =   0;
```

Signal ATBUFSEL* is active when DSP U1 addresses the DSP buffer, signal ATINT is the port processor interrupt (DSP U1 may interrupt port processor U87 by asserting this line, and DSP U1 may set and reset signal ATINT), signal ATRAMSEL* is active when DSP U1 addresses

its program RAM units (U54 and U67), and signal ATROMSEL* is active when DSP 1 addresses its boot ROM units (U7 and U67).

DSP U1 has access to RAM units U54 and U67, each RAM unit having capacity to store 8K eight-bit words. In the Fig. 4 embodiment, DSP U1 has a memory address space with only 4K capacity. Accordingly, DSP U1 has access to two banks of RAM, which are selectable by port processor U87. Programs are loaded into RAM units U54 and U67 by DSP U1 under the control of port processor U87.

DSP U1 also has access to ROM units U2 and U7, each of which is preferably a PROM with capacity to store 512 eight-bit words. ROM units U2 and U7 preferably contain boot code to test DSP subsystem 110, to download code from port processor U87 and upload code to port processor U87, and to execute code in program RAM.

Port processor U87 communicates with DSP U1 through a 256 byte data buffer, comprising 2K x eight-byte RAM unit U80, programmable logic unit U39, an eight-bit counter including devices U75 and U81, data buffers U46 and U66, and data register U53. Unit U39 provides timing and control for unit U80 and all control signals for the data buffer between port processor U87 and DSP U1, and is addressed as a single location in the address space of both processors U87 and U1. The address for buffer RAM U80 is provided by an eight bit counter, preferably comprising two synchronous four-bit counters U75 and U81 (which may be Texas Instruments Inc. SN74S163 integrated circuits) connected as shown in Figure 4. This counter may be explicitly loaded by port processor U87, and will increment with every buffer access. A series of reads from the same location (by either processor U87 or U1) will read a series of sequential data bytes in buffer RAM U80. Port processor

U87 may also read up to four sequential locations, starting with [AMA2, AMA1, AMA0] = [0, 0, 0], to retrieve four sequential bytes from buffer RAM U80.

The buffer comprising devices U80, U39, U75, U81, U46, U66, and U53 provides a purely simplex communications path between processors U87 and U1. Only one of processors U87 and U1 may access the buffer at a time. Typically, one processor will fill the buffer, signal the other processor, and then wait for a signal before accessing the buffer again.

Circuit U39 generates the following signals: ACNTLD*, the load signal to the buffer address counters U75 and U81 (processor U87 may load an address into the counters by writing to the buffer address space with AMA2 high); ACNTCLK, the clock signal to address counters U75 and U81 (A rising edge will occur at the end of every access by either processor U87 or U1, incrementing the counter. The counter will also be clocked during a load operation); ATBRD*, when low, will enable data buffer U46 to DSP U1; ATBWR*, when low, will enable register U53 during a DSP write to the buffer (the buffer is loaded from the data bus connecting DSP U1 with U46 and U53, on every rising edge of signal ATCLK); AMBEN*, which enables data buffer U66 to port processor U87 (the direction of this buffer is controlled by signal AMRW); ARAMWR*, the write signal to RAM U80 (which is active whenever processor U87 or U1 writes to RAM U80); and ARAMEN*, which enables RAM U80 outputs, and will be active whenever DSP U1 or port processor U87 reads from RAM U80.

Table 14 specifies the logical relationships between the input and output signals of device U39:

Table 14

```
!CNTLD    =    !AXCNTLD;
!ACNTCLK  =    (((E & !AMBUFSEL # !AMBFN) # !ATBWR) #
               !ATDEN & !ATBUFSEL);
!ATBRD    =    !ATDEN & !ATBUFSEL;
!AXCNTLD  =    (E & !AMBUFSEL & AMA2 & !AMRW #AXCNTLD &
               !ACNTCLK);
!ATBWR    =    (!ATBUFSEL & !ATWEN # !ARAMWR & !ATBWR);
!AMBEN    =    (E & !AMBUFSEL & !AMA2 # !ARAMWR &
               !AMBEN);
!ARAMWR   =    (E & !AMBUFSEL & !AMA2 & !AMRW #
               !ATBUFSEL & !ATWEN);
!ARAMEN   =    (E & !AMBUFSEL & !AMA2 & AMRW # !ATBUFSEL
               & !ATDEN);
```

Device U39 is designed to maintain a strict precedence relation among several pairs of signals, in order to guarantee hold times when required. In the following pairs of signals, a rising edge on the first will always precede a rising edge on the second: ARAMWR*, ATBWR*; ATBWR*, ACNTCLK; ARAMWR*, AMBEN*; AMBEN*, ACNTCLK; and ACNTCLK, ACNTLD*.

Figure 5 is a schematic diagram of interface unit 114 of PCI unit 3. Unit 214 is identical to unit 114, and is interconnected with units 206, 208, 210, 212, 102, and 104 in the same manner as unit 114 is interconnected with units 106, 108, 110, 112, 102, and 104. Accordingly, a separate diagram of unit 214 would duplicate Figure 5, and is not provided.

Interface unit 114 includes CODEC U104, which preferably is a Model TP3051 parallel access mu-Law CODEC (available from National Semiconductor) capable of digitizing incoming analog signals at the rate of 8000 eight-bit samples per second. All timing and control for interface unit 114 is generated by counter U82 and programmable logic device U83 (shown in Figure 3). In

the Fig. 3 embodiment, the control signals output by device U83 not only serve to control CODEC U104 of interface 114, but the corresponding identical CODEC of interface 214. Counter U82 is preferably a Texas Instruments Inc. SN74393 integrated circuit, and device U83 is preferably a PAL16R6 integrated circuit.

The output signals generated by device U83 are: CDCLK and S2, two state variables output by a "divide by four" counter (the clock for device U83 has frequency 4MHz, so that signals CDCLK and S2 have frequency 1MHz), with signal CDCLK used by CODEC U104 to drive its switched capacitor filters, and by device U82 to create a 125 microsecond sampling period; CDXFER*, which indicates to CODEC 104 that a read or write transfer is about to take place (two pulses of signal CDXFER* occur every 125 microsecond); LATCHOUT, which clocks data from CODEC 104 into register U95 (shown in Figure 5) once every 125 microsecond during the data transfer process; LATCHIN*, which enables data from register U91 (shown in Fig. 5) onto the CODEC data bus (port processor U87 loads the data into register U91, and the data enabled onto the CODEC data bus is loaded onto CODEC U104); FIRQ*, which sets an RS latch which issues a fast interrupt request to port processor U87 (FIRQ* goes low every 125 microseconds); and CNTRST, the inverse of FIRQ*, which is used to asynchronously reset the external counter.

Signals CDXFER*, LATCHOUT, LATCHIN*, FIRQ*, and CNTRST are outputs of a state machine described by the state transition table shown in Table 15:

Table 15

	PRESENT STATE								NEXT STATE							
	VRESET	beg in	CDXFER*	LATCHOUT	LATCHIN*	FIRQ*	CNTRST		CDXFER*	LATCHOUT	LATCHIN*	FIRQ*	CNTRST	CDCLK	S2	
1	1	X	X	X	X	X	X		1	1	1	1	0	X	X	
2	0	0	1	1	1	1	0		1	1	1	1	0	X	X	
3	0	1	1	1	1	1	0		0	1	1	0	1	0	0	
4	0	X	0	1	1	0	1		0	1	1	0	1	1	0	
5	0	X	0	1	1	0	1		1	1	0	0	1	1	1	
6	0	X	1	1	0	0	1		1	1	0	0	1	0	1	
7	0	X	1	1	0	0	1		1	1	1	0	1	0	0	
8	0	X	1	1	1	0	1		1	1	1	0	1	1	0	
9	0	X	1	1	1	0	1		0	1	1	0	1	1	1	
10	0	X	0	1	1	0	1		0	0	1	0	1	0	1	
11	0	X	0	0	1	0	1		1	0	1	0	1	0	0	
12	0	X	1	0	1	0	1		1	1	1	1	0	1	0	

Table 16 specifies the logical relationships between the input and output signals of device U83:

Table 16

```
!CDCLK = (S2 # VRESET);
!S2 = (!CDCLK # VRESET);
!CDXFER = (((!CNT7 & CNT6 & CNT5 & CNT4 & CNT3 & !CNT2 &
      CNT1 & CNT0 & !VRESET # !CDCLK & !S2 & !CDXFER &
      LATCHOUT & LATCHIN & !FIRQ & !VRESET) # !CDCLK &
      !S2 & CDXFER & LATCHOUT & LATCHIN & !FIRQ &
```

```
        !VRESET) # CDCLK & S2 & !CDXFER & LATCHOUT &
        LATCHIN & !FIRQ & !VRESET);
!LATCHOUT = (CDCLK & S2 & !CDXFER & LATCHOUT & LATCHIN &
        !FIRQ & !VRESET # !CDCLK & S2 & !CDXFER &
        !LATCHOUT & LATCHIN & !FIRQ & !VRESET);
!LATCHIN = (CDCLK & !S2 & !CDXFER & LATCHOUT & LATCHIN &
        !FIRQ & !VRESET # CDCLK & S2 & CDXFER &
        LATCHOUT & !LATCHIN & !FIRQ & !VRESET);
!FIRQ = ((((!CNT7 & CNT6 & CNT5 & CNT4 & CNT3 & !CNT2 &
        CNT1 & CNT0 & !VRESET # !S2 & LATCHOUT & LATCHIN
      & !FIRQ & !VRESET) # S2 & CDXFER & LATCHOUT &
  !LATCHIN & !FIRQ & !VRESET) # CDCLK & S2 &
!CDXFER & LATCHOUT & LATCHIN & !FIRQ & !VRESET);
!CDCLK & S2 & !CDXFER & !LATCHOUT & LATCHIN &
!FIRQ & !VRESET);
!CNTRST = FIRQ.
```

Port processor U87 writes to CODEC 104 by loading CODEC input register U91 with PCM or control data following the 8KHz interrupt (signal FIRQ). At the next 8KHz interrupt, this data is then loaded into CODEC 104. At the same time, data may be transferred from CODEC output buffer U95. Data in buffer U95 may be read at any time by port processor U87.

The analog input stage of CODEC 104 (in its Model TP3051 embodiment) is configured as an inverting amplifier. Inputs VFXI+ and VFXI- are the non-inverting and inverting inputs, respectively, to an op-amp within of CODEC 104. Signal GSX is the feedback output of the op-amp. VFXI+ is grounded, and 22,100 Ohm feedback resistor R89 and 22,100 Ohm input resistor R88, set the input gain to equal -(R89)/R88 = -(1). With such input gain, the maximum signal level allowed at the input of the inverting amplifier configuration, measured at R88, is (+/-)2.5 V.

The analog output from CODEC 104 is signal VFRO. This output is capable of driving a 600 ohm load to (+/-)2.5 V. Line VFRO is capacitively coupled to hybrid circuit 200 (to be discussed below) via a 0.1 microfarad capacitor C94.

Hybrid circuit 200 includes two op-amps U103 and passive components as shown in Fig. 5. The purpose of circuit 200 is to merge the transmitted signal from CODEC U104 onto the two wire telephone pair and to extract the received signal (to be forwarded to CODEC U104) from the two wire telephone pair. In order to approximately match the full range of typical telephone line impedances, hybrid circuit 200 shown in Fig. 5 is designed to offer greater than 20dB of transhybrid loss over the spectrum of typical telephone lines.

Received signals from the telephone interface are present at the secondary of isolation telephone interface transformer T1. The transmitted voice signal from CODEC U104 is fed to a gain stage comprising the upper op-amp U103 in Fig. 4, resistor R85, capacitor C90, and resistor R83. The output of this gain stage drives resistor R77 and telephone interface transformer T1.

The received signal at the secondary of transformer T1 is fed to the gain stage comprising the lower op-amp 103 in Fig. 5, resistors R78 and R79, capacitor C86, and resistor R82. Received voice signals are transmitted through capacitor C91 to CODEC U104, and received DTMF signals are transmitted through capacitor C78 to DTMF detection device U102 to be discussed below.

Interface 114 is capable of interfacing with Loop, Reverse Loop, and E&M telephone lines. In order for the telephone interface to satisfy the FCC dielectric strength requirement that less than 10mA flow to ground when 1000 or 1500 VAC is applied across any and all

conductive points external to the system, the following components of interface 114 should be selected from those commercially available which meet such dielectric strength standard: transformer T1, optoisolators U107, U100, U106, U105, U02, U96, and U108, relays K3 and K4, and switches S1, S2, and S3.

V130LA5 Metal Oxide Varistors RV3 and RV4 are provided between ground and the Ring and Tip lines, respectively, to ensure that interface 114 meets the FCC surge and lightning protection requirements. For the same reason, V68ZA2 Metal Oxide Varistors RV1 and RV2 are provided between ground and the EAR and MOUTH lines, respectively.

The audio path is always connected to the Tip and Ring lines, through transformer T1 and capacitor C100. The DC path can be switched between Tip/Ring and Ear/Mouth by means of switch S3. Relays K3 and K4 can be operated independently to control On-Hook and Off-Hook conditions. The termination block comprises switches S1 and S2, termination impedances, and battery, ground, and current sensors. Each current sensor includes an opto-isolator.

Capacitor C102 and resistor R90 are connected in series with optoisolator U108. In parallel with this circuit is transformer T1 in series with capacitor C100. Resistor R90 and capacitor C102 provided a desired amount of current through optoisolator U108 during ringing. The current through U108 is converted to a voltage for ring signal detection. When current is not flowing through U108, resistor R93 pulls terminal 5 of device U108 high. When a correct ring voltage is being detected, U108 will begin conducting between terminals 4 and 5. Since pin 4 of U108 is grounded, the voltage at terminal 5 of U108 is pulled low. When the voltage at terminal 5 of U108 reaches a TTL logic low, pin 5 of

buffer U93 (line ARI) goes low. The signal on line ARI is supplied to VIA U77 (shown in Figure 4).

PCI unit 3 is designed to meet the requirements of the Loop Start, Ground Start, DID, and E&M telephone signalling standards, and may be connected to either end of a telephone line (the station or office end). Switches S1, S2, and S3 are controllable to provide the necessary hardware interface for any of the configurations mentioned in the preceding sentence, as shown in Table 17:

Table 17

	Switch Positions		
	S1	S2	S3
Loop Start			
Station End	A	A	A
Office End			
(Battery Interrupt)	B	A	A
(Battery Reversal)	B	B	A
Ground Start			
Station End	A	B	A
Office End	B	A	A
DID			
Station End	B	B	A
Office End	A	A	A
E&M Type I			
(Side A)	B	B	B
(Side B)	B	A	B

Octal port U85 (discussed above with reference to Figure 4) receives switch status signals ASW1-ASW3 from switches S1, S2, and S3, respectively.

Signalling relays K3 and K4 are driven by transistor drivers Q4 and Q3, respectively. Relays K3 and K4 are controlled by programmable logic device U89. During normal operation, signal A6809 is high, giving

control of the relays to port processor U87 via signals ARLY1 and ARLY2. When signal A6809 is low, control of the relays is given to the VME bus master (CPU 1). The VME bus master may set the relays to a desired state determined by the state of signal AOFFH and the state of switches S1, S2, and S3.

As an example, operation of the phone line interface will be described for the case that the positions of switches S1, S2, and S3 have been set in positions A, A, and A, respectively, for Loop Start (Station End) operation. In this configuration, when "On hook", both relays K3 and K4 are deenergized (in their "zero" position), and when PCI unit 3 initiates a call (i.e., goes "Off-hook") relays K3 and K4 are energized (in their "one" position). Optoisolators U106 and U105 are employed to determine distant end status. When the distant end is On-hook, and if Tip and Ring are connected correctly, line AFLOOP (connected to terminal 5 of device U105) will be conducting (On), and line ARLOOP (connected to terminal 5 of device U106) will not be conducting (Off). Table 18 indicates the status of lines ARLOOP and AFLOOP under various distant end conditions when the local end (PCI unit 3) is Off-hook:

Table 18

	Tip and Ring Reversed		Tip and Ring Normal	
	ARLOOP	AFLOOP	ARLOOP	AFLOOP
Off-hook	0	1	1	0
Battery Reversal CO On-hook	1	0	0	1
Battery Interruption CO On-hook	1	1	1	1
Normal CO On-hook	0	1	1	0

In Table 18, the abbreviation "CO" refers to the central office.

Table 19 specifies the logical relationships between the input and output signals of device U89:

Table 19

```
ARCTL1 = !(A6809 & !ARLY1 # !A6809 & !AOFF & !ASW2
& !ASW3 # !A6809 & !AOFF & ASW1 & ASW3 # !A6809 & ASW1 &
!ASW2 & ASW3 # !A6809 & AOFF & ASW1 & ASW2 & !ASW3);
ARCTL2 = !(A6809 & !ARLY2 # !A6809 & !AOFF & !ASW1
& !ASW3 # !A6809 & !AOFF & !ASW2 & !ASW3 # !A6809 &
!AOFF & ASW1 & ASW3 # !A6809 & AOFF & ASW1 & ASW2).
```

The loop current through the telephone line must be accurately determined to facilitate distant end Off-hook detection. This detection operation must occur over a range of central office (CO) battery voltages and a range of loop resistances. The loop current is monitored by VIA U77 (shown in Figure 4), by monitoring the state of signals AFLOOP and ARLOOP, which are sent from optocouplers U105 and U106 to VIA U77.

Interface unit 114 of PCI unit 3 includes a dedicated dual tone multi-frequency (DTMF) detector, comprising a bandsplit filter U102 (preferably an AMI 3525 bandsplit filter) for separating the two tone groups used for DTMF signalling, and a DTMF decoder U101 (preferably a Mostek 5103 DTMF decoder). Crystal clock Y3 supplies a 3.58 MHz clock signal to oscillator terminals 16 and 17 of U102, for internal use within U102 and for generating the CKOUT clock signal which emerges form terminal 18 of U102. In an embodiment including a second port (Port B), a counterpart to clock Y3 need not be included in such second port. Rather, signal CKOUT may be supplied to the oscillator terminals of such second port's counterpart to device U102.

DTMF decoder U101 generates an interrupt to port processor U87 whenever a valid DTMF digit is detected, and this interrupt is transmitted to device U87 on line ADTMF. When a valid digit is detected, the data is latched into output buffer U65, and does not change until the next valid digit is detected. Port processor U87 may read the output data by addressing output buffer U65 (which preferably is a Texas Instruments SN74S299 register).

It is possible to connect the audio path from one port to the other using audio loopback relay K5 and the associated passive circuitry (shown in Figure 3). Relay K5 is preferably a DPST relay connecting both audio paths to the passive network comprising transistor Q5, which provides about 30 dB attenuation from one port to the other. Relay K5 may be controlled by port processor U87.

Figures 2-5 include references to "Sheets 1-6". Sheet 1 corresponds to Figure 2, sheet 2 corresponds to Figure 3, sheet 3 corresponds to Figure 4, sheet 4 corresponds to a counterpart to Figure 4 (which counterpart would be identical to Figure 4 except that it would show port processor 206 and DSP 210 of Port B, rather than port processor 106 and DSP 110 of Port A), sheet 5 corresponds to Figure 5, and sheet 6 corresponds to a counterpart to Figure 5 (which counterpart would be identical to Figure 5 except that it would show interface 214 of Port B, rather than identical interface 114 Port A).

A preferred embodiment of CPU 1 will next be described with reference to Figures 6 and 7. CPU 1 performs all system processing for the inventive voice processing system. CPU 1 appears to VME bus 7 as a master with 16 bit and 8 bit data transfer capability and 24 bit addressing capability. The circuitry

supporting the VME bus supervisory functions includes system clock driver 15, bus arbiter 17, IACK daisy chain driver 19, and bus timer 21.

Address decoding is implemented with to programmable logic devices U107 and U300. Device 107 generates the following signals: VM*, which is active when accessing a device on VME bus 7; SHORT*, which is active when addressing the short address space of VME bus 7 this signal is used to change the VME address modifier to indicate that the current access uses only 16 bits of address; ROM*, which is active when accessing ROM units U302 and U502; MMU*, which is the chip select for memory management unit 31; DUART, which is the chip select for DUART U207; PARA, which is the chip select for parallel interface and timer unit U206; SCSI*, which is the chip select for the SCSI subsystem; and UART*, which is active when accessing any of several devices configured as "VPA" devices, including the modem, the real time clock, the floppy disk subsystem reset, and the VME bus release. Table 20 specifies the logical relationships between the input and output signals of device U107:

TABLE 20

```
UART  =  !(!MALL & !PA12 & !PA13 & PA14 & PA15 & PA16 & PA17 & PA18
         & PA19 & PA20 & PA21 & PA22 & PA23 & !PAS);

SCSI  =  !(!MALL & PA12 & PA13 & !PA14 & PA15 & PA16 & PA17 & PA18 &
         PA19 & PA20 & PA21 & PA22 & PA23 & !PAS);

PARA  =  !(!MALL & !PA12 & PA13 & !PA14 & PA15 & PA16 & PA17 & PA18
         & PA19 & PA20 & PA21 & PA22 & PA23 & !PAS);

DUART =  !(!MALL & PA12 & !PA13 & !PA14 & PA15 & PA16 & PA17 & PA18
         & PA19 & PA20 & PA21 & PA22 & PA23 & !PAS);

MMU   =  !(!MALL & !PA12 & !PA13 & !PA14 & PA15 & PA16 & PA17 & PA18
         & PA19 & PA20 & PA21 & PA22 & PA23 & !PAS);
```

```
ROM     =   !(!MALL & !PA17 & !PA18 & !PA19 & !PA20 & !PA21 & !PA22 &
            !PA23 & !PAS);

SHORT   =   !(!MALL & !PA15 & PA16 & PA17 & PA18 & PA19 & PA20 & PA21 &
            PA22 & PA23 & !PAS);

VM      =   !(!PA22 & PA23 & !PAS
              # PA22 & !PA23 & !PAS
              # !PA21 & PA23 & !PAS
              # PA21 & !PA23 & !PAS
              # !PA20 & PA23 & !PAS
              # PA20 & !PA23 & !PAS
              # !PA19 & PA23 & !PAS
              # PA19 & !PA23 & !PAS
              # !PA18 & PA23 & !PAS
              # PA18 & !PA23 & !PAS
              # !PA17 & PA23 & !PAS
              # PA17 & !PA23 & !PAS
              # !PA16 & PA23 & !PAS
              # !PA15 & PA23 & !PAS)
```

Device U300 generates device select signals for all the eight bit peripherals ("VPA" devices) which are accessed in the same manner as 6800 peripherals (i.e., synchronous with the E clock). Each such acces cycle is initiated when VPA* is asserted to system processor U201 (which is preferably a Motorola MC68010 integrated circuit) in response to a valid address. Processor U201 will then assert signal VMA in response. Each transfer to or from the peripheral must be synchronous with clock E.

Device U300 generates the following device select signals: MODEM*, which is the chip select for modem controller U303; RTC*, which is the chip select for real time clock U604; MRST*, which is the reset for the floppy disk subsystem controller (this is one output of a latch which preferably may be set or reset under software control); and MRLS*, which releases VME bus 7 under software control (this is also an output of a latch which preferably may be set or reset under software control. Table 21 specifies the logical

relationships between the input and output signals of device U107:

TABLE 21

```
MODEM  =   !(!UART & !A6 & !A7);
RTC    =   !(!VMA & A6 & !A7);
MRST   =   !(!(!UART & !A6 & A7 & A1) & NMRST & RST);
NMRST  =   !(!(!UART & !A6 & A7 & !A1) & MRST # !RST);
MRLS   =   !(!(!UART & A6 & A7 & A1) & NMRLS & RST);
NMRLS  =   !(!(!UART & A6 & A7 & !A1) & MRLS # !RST);
BMRST  =   !NMRST;
```

Control signals are generated by programmable logic device U301 and logic devices 201, 202, and 203. Device U301 generates the following signals: VME*, which is active whenever a VME bus access is requested by system processor U201 or when a VME bus interrupt request is being acknowledged; PAS*, which is a physical access strobe signal, and is active when address strobe AS* from processor U201 and address strobe MAS* from memory management unit U202 (which is preferably a Motorola MC68451 integrated circuit) are both active; DSI, which is active when either the upper or the lower data strobe is active; RIN, which is used to qualify the data strobes to create the physical data strobes, and is active when RW* is low (indicating a write cycle) and WIN* (the write inhibit from U202) is low (indicating an attempt to write to a protected segment); and ^{B}ERRST, which resets the bus error timer (the timer begins counting when a bus cycle begins, indicated by AS*, DS0*, or DS1* going low, and is reset when all of these

signals go high). Table 22 specifies the logical relationships between the input and output signals of device U301:

TABLE 22

```
VME   =   !(VIACK # !VM);
PAS   =   !(!AS & !MAS);
DSI   =   (!LDS # !UDS);
RIN   =   (!RW & !WIN);
BERRST =  !(!AS & RST # !DS0 & RST # !DS1 & RST);
```

Logic devices 201, 202, and 203 generate the following signals: address strobe MAS*, which is generated by open collector driver 201 coupled to line MALL emerging form U203 (when signal MAS* is driven low, device U202 is prevented from performing an address translation. This occurs during an interrupt acknowledge); ALL, which is also generated by open collector driver 201 (signal ALL is driven low during an interrupt acknowledge to prevent U202 from interpreting

the failure to translate the address as a fault); PLDS*, which is generated by circuit 202 coupled to the control bus of VME bus 7, and prevents a write to protected memory (PLDS* consists of data strobe LDS* qualified by signal RIN); PUDS*, which is also generated by circuit 202; and VPA*, which is generated by circuit 203, and is a valid peripheral address signal indicating to processor U201 that the current cycle is either an access to a 6800 type peripheral, or an autovectored interrupt (in the first case, the data transfer is synchronous to clock E and requires no DTACK* in response; in the second case, processor U201 does not require an externally generated interrupt vector and instead processor U201 uses a dedicated vector for the current interrupt level).

VME bus 7 is asynchronous, so that all transfers require handshaking between the VME bus master (CPU 1) and a slave (each PCI unit 3). The handshaking signals employed in a data transfer are: AS* (address strobe), DS0* and DS1* (data strobe signals), DTACK* (a data transfer acknowledgement signal, the inverse of DTACK), and BERR* (a bus error signal). During a data transfer or interrupt acknowledge, CPU 1 drives AS*, DS0*, and DS1*.

All bus control signals are generated by VME bus controller U103 (which preferably is a Model 68172 integrated circuit). Circuit U103 is configured for single master operation, and performs the following functions: 1. It requests access to VME bus 7 when processor U201 addresses the VME bus address space. The onboard processor signals this by bringing VME* and MAS* low. If the bus master does not own bus 7, circuit U103 signals its request by driving BR* low. The bus arbiter (U006) grants this request by driving BGIN* low. U103 then responds by driving BBSY* low and then releasing

BR*. CPU 1 nay then access bus 7 normally, If CPU 1 already has ownership of bus 7, bus 7 may be accessed immediately; 2. It releases bus 7 on request. If another bus master requests access to bus 7, bus arbiter U006 will issue a clear command by driving BCLR* low. This line will interrupt CPU 1, which may schedule a time to release bus 7. MRLS* is driven low to signal a bus release to arbiter U006. Arbiter U006 drives RELSE high, signalling U103 to release the bus. U103 does so by releasing BBSY*; 3. It initiates and completes a data transfer by driving VME* and PAS* low. U103 generates the VME bus data strobe (VAS*) and waits for VDTACK* or VBERR* from the VME bus slave. U103 also generates the control signals for the VME bus buffers (to be described below). Responses from the VME bus are regenerated onboard by U103 as LDTACK* and LBERR*.

In addition to controlling and monitoring the VME bus handshake signals directly, U103 generates the following control signals for the VME bus interface: DEN*, which is a data enable for enabling the VME bus buffers (this signal goes low when the VME bus is selected (i.e., VME* and PAS* are low) and DSI is high); DSEN*, which is a data strobe enable, which when low, passes the data strobes to bus 7 (this line will be active when DSI, VDTACK*, and BERR* are high, and VAS* is low); VMEEN*, which is a buffer tri-state control signal which enables the address lines and address modifiers onto the VME bus (it is low whenever CPU 1 owns bus 7); and $D^{D}IR$, which is a data direction signal which determines the direction of the VME bus data buffers, as specified by signal R/W*.

Bus arbitration is on a fixed priority basis, with level three highest and level zero lowest. Arbitration is accomplished by two state machines implemented by programmable logic device U006: one to grant the bus in

response to a bus request; the other to release the bus in response to a request form CPU 1 to release the bus. The functions of the two state machines are described in Table 23:

TABLE 23

```
State_Diagram release_bus

  State HOLD_BUS:     if !RST then HOLD_BUS
                      else if !MRLS then RELSE_BUS
                      else HOLD_BUS;

  State RELSE_BUS:    if !RST then HOLD_BUS
                      else if BBSY then HOLD_BUS
                      else RELSE_BUS;

State_Diagram grant_bus

  State IDLE:         if (!BBSY # !RST) then IDLE
                      else if !BR3 then GRANT3
                      else if !BR2 then GRANT2
                      else if !BR1 then GRANT1
                      else if !BR0 then GRANT0
                      else IDLE;

  State GRANT3:       if (BBSY & RST) then GRANT3 else IDLE;

  State GRANT2:       if (BBSY & RST) then GRANT2 else IDLE;

  State GRANT1:       if (BBSY & RST) then GRANT1 else IDLE;

  State GRANT0:       if (BBSY & RST) then GRANT0 else IDLE.
```

With reference to Table 23, if bus 7 is under the control of a bus master, indicated by an active level on BBSY*, RELSE is inactive. If the current bus master is CPU 1, it will continue to hold bus 7 until it releases it under software control by bringing MRLS* low. RELSE will go active, causing the VME bus interface to release the bus by bringing BBSY* high. The "grant bus" state machine remains in an idle state until a bus request is received. When simultaneous requests are received, bus 7 is granted to the highest priority request. Bus 7

remains under control of the grantee until released by bringing BBSY* high. BCLR* is active whenever a bus request is active, and at that time generates a level six interrupt request to CPU 1. CPU 1 may respond to the bus request at any time after the request is made.

CPU may be configured at any of four priority levels by setting jumpers JP1, JP6, and JP7.

Circuit U006 also functions as the VME bus IACK daisy chain driver. Signal IACKOUT* is active when an interrupt acknowledge cycle is underway, as indicated by an active IACKIN* signal. IACKOUT* is qualified with DS0* and VAS*, and is delayed by the system clock SYSCLK, to meet the VME bus specification requiring a 40ns delay from the falling edge of DS0* to the falling edge of IACKOUT*.

Table 24 specifies the logical relationships between the input and output signals of device U006:

TABLE 24

```
RELSE: =    !(!RST # BBSY & RELSE # MRLS & !RELSE);

BG3OUT:=    !(BBSY & BG0OUT & BG1OUT & BG2OUT & !BG3OUT & RST # BBSY &
            BG0OUT & BG1OUT & BG2OUT & !BR3 & RST);

BG2OUT:=    !(BBSY & BG0OUT & BG1OUT & !BG2OUT & BG3OUT & RST # BBSY &
            BG0OUT & BG1OUT & BG3OUT & !BR2 & BR3 & RST);

BG1OUT:=    !(BBSY & BG0OUT & !BG1OUT & BG2OUT & BG3OUT & RST # BBSY &
            BG0OUT & BG2OUT & BG3OUT & !BR1 & BR2 & BR3 & RST);

BG0OUT:=    !(BBSY & !BG0OUT & BG1OUT & BG2OUT & BG3OUT & RST # BBSY &
            BG1OUT & BG2OUT & BG3OUT & !BR0 & BR1 & BR2 & BR3 & RST);

IACKOUT:=   !(!DS0 & !IACKIN & !VAS);

BCLR:  =    !(!BR0 # !BR1 # BR2 # !BR3).
```

CPU 1 includes system clock unit 204, including two crystal oscillators, each implemented with two gates of U601. The frequency of oscillation is twice the required value, and D flip-flop U701 divides each oscillator's

output frequency by two to ensure a fifty percent duty cycle. The first oscillator has a 20MHz frequency, divided down to 10MHz, and serves as the system clock SYSCLK (used by circuits U201, U202, U103, U206, and U208). The second oscillator has a 32MHz frequency, divided down to 16MHz, and serves as the VME bus clock, which drives all PCI units 3 coupled to bus 7. Either or both of the oscillators may be disabled (to facilitate automated testing) by driving the pulled up inputs to NAND gates U601 low.

Dual four-bit counters U603 and U708 (which are cascaded to form a 16-bit counter, and are preferably Texas Instruments Inc. 74LS393 integrated circuits) serve as a timer for monitoring bus activity. Counters U603 and U708 are driven by the 1 MHz E clock from processor U201. If signal AS*, UDS*, or LDS* goes low, BERRST goes low, allowing the counters to start counting. The counter will continue to count until all of these signals go high. A bus error occurs if the most significant bit goes high.

The VME reset circuit 206 comprising timer U703 and external timing components (resistors R13 and R20, capacitors C19 and C21, and a pair of inverters U602) will cause a system-wide reset on power-up, or when the operator actuates a reset switch SW1. Following power-up, C21 will discharge, holding the trigger and threshold inputs of timer circuit U703 at ground, and forcing circuit U703's output high. The output of U703 is inverted by one inverter U602, which holds RST* low. The entire system is reset by line RST*, via VME bus reset driver U500. C21 will discharge through R13, preferably with time constant 470 ms. When the voltage on C21 exceeds the threshold of timer U703 (typically 516 ms), timer U703 will reset, releasing RST*. Both RST* and HALT* are held low at power-up as required by

processor U201 for a complete reset. The operator may halt processor U201 by actuating halt switch SW2.

System processor U201 is preferably a Motorola 68010 integrated circuit, and memory management unit U202 is preferably a Motorola 68451 integrated circuit. A Motorola 68010 integrated circuit is a 32 bit processor with a 16 bit external data bus, D0-D15, and a 24 bit address bus, A1-A23, plus LDS* and UDS*, and may access individual bytes by asserting both LDS* and UDS*, or 16 bit words by asserting either LDS* or UDS*. Lower order bytes, addressed by LDS* are at odd logical addresses, while higher order bytes, addressed by UDS*, are at even addresses. Words are at even addresses.

Processor U201, in an embodiment in which it is a Motorola 68010 integrated circuit, is an asynchronous processor, so that every memory or peripheral access (with the exception of a VPA access, described elsewhere herein) requires a response on line DTACK*. DTACK* may be generated by the addressed device (such as U202, U206, DUART U207, or the VME bus slave), or by external circuitry (as in the case of the SCSI controller subsystem including U208), or by the ROM. The remaining devices assert VPA when addressed.

If U201 receives a bus error, indicated by BERR* low, instead of DTACK*, U201 will enter an exception processing routine to resolve the error, in which U201 halts execution and drives HALT* low. Processor U201 must be reset to exit this state.

Every access by U201 includes a function code, FC0-FC2, which defines the type of access. The function codes is also passed on to the VME bus in the form of an address modifier. These codes are set forth in Table 25:

TABLE 25

FC[2:0]	SHORT	AM[5:0]	Definition
000	0	$28	Undefined
001	0	$29	Short address user data
010	0	$2A	Undefined
011	0	$2B	Undefined
100	0	$2C	Undefined
101	0	$2D	Short address supervisor data
110	0	$2E	Undefined
111	0	$2F	Reserved (Interrupt acknowledge)
000	1	$38	Undefined, reserved
001	1	$39	Standard address user data
010	1	$3A	Standard address user program
011	1	$3B	Undefined, reserved
100	1	$3C	Undefined, reserved
101	1	$3D	Standard address supervisor data
110	1	$3E	Standard address supervisor program
111	1	$3F	Reserved (Interrupt acknowledge)

The multiple master capability of the 68010 integrated circuit (using lines BR*, BG*, and BGACK*) is not utilized in the preferred embodiment described with reference to Figs. 6 and 7. These lines are preferably unconnected or pulled to an inactive state.

The purpose of memory management unit U202 is to translate the logical addresses form U201 to physical addresses used by the system. This allows U201 to share code, while maintaining separate data spaces.

All memory references in programs executed by U202 will be logical addresses, which appear on lines A1-A23 from U201. Based on function code FC0-FC2, U202 translates the logical address to a physical address in any of four different memory spaces: User data (corresponding to [FC0, FC1, FC2]=001), User program (corresponding to [FC0, FC1, FC2]=010), Supervisor data

(corresponding to [FC0, FC1, FC2]=101), and Supervisor program (corresponding to [FC0, FC1, FC2]=110). The signal [FC0, FC1, FC2]=111 indicates and interrupt acknowledge. The FC3 input to U202 is not used, and is tied to ground.

All registers in memory management unit U202 are programmed by U201 following power-up. U201 and U202 communicate via a bidirectional, dual purpose bus AD0-AD15. During normal translation, the physical address appears on this bus, latched by octal D-latches U104 and U105. The gating signal is HAD*, generated by U202. When U201 accesses U202 as a peripheral, AD0-AD15 is used as a data bus. Two bidirectional buffers, U401 and U501, isolate AD0-AD15 and D0-D15> Buffers U401 and U501 are enabled by ED*, generated by U202. The direction of the buffers is controlled by R/W*.

Circuit U202 supports seven prioritized interrupt levels. Interrupt requests are encoded on IPL2* through IPL0*, and INTIRQ*, by programmable logic device U009. Device U009 generates the interrupt acknoledge signals for all system interrupts. These fall into three categories: external vectored interrupts (VIACK); internal vectored interrupts (MIACK*, TIACK*, PIACK*, and DIACK*); and internal autovectored interrupts (AUTO*). Internal interrupts are indicated by INTIRQ* being active (low). Each autovectored interrupt is separately decoded. All other interrupts are assumed to be external VME bus interrupts. The priority rules are: First, higher numbered requests have higher priority; and Second, external (VME bus) requests have higher priority.

Table 26 specifies the logical relationships between the input and output signals of device U009:

TABLE 26

```
IPL2    =    !(!ACFAIL
              # IR7
              # !IRQ7
              # !BCLR
              # !IRQ6
              # !IR5
              # !IRQ5
              # !IR4
              # !IRQ4);

IPL1    =    !(!ACFAIL
              # IR7
              # !IRQ7
              # !BCLR
              # !IRQ6
              # !IR3 & IR4 & IR5 & IRQ4 & IRQ5
              # IR4 & IR5 & !IRQ3 & IRQ4 & IRQ5
              # !IR2 & IR4 & IR5 & IRQ4 & IRQ5
              # IR4 & IR5  !IRQ2 & IRQ4 & IRQ5);

IPL0    =    !(!ACFAIL
              # IR7
              # !IRQ7
              # BCLR & !IR5 & IRQ6
              # BCLR & !IRQ5 & IRQ6
              # BCLR & !IR3 & IR4 & IRQ4 & IRQ6
              # BCLR & IR4 & !IRQ3 & IRQ4 & IRQ6
              # BCLR & !IR1 & IR2 & IR4 & IRQ2 & IRQ4 & IRQ6
              # BCLR & IR2 & IR4 & !IRQ1 & IRQ2 & IRQ4 & IRQ6);

INTIRQ  =    !(!A1 & !A2 & A3 & IRQ4
              # !A1 & A2 & !A3 & IRQ2
              # !A1 & A2 & A3 & IRQ6
              # A1 & !A2 & !A3 & IRQ1
              # A1 & !A2 & A3 & IRQ5
              # A1 & A2 & !A3 & IRQ3
              # A1 & A2 & A3 & IRQ7).
```

Interrupt acknowledge signals are generated by programmable logic device U203. Each such signal informs the interrupting source that the current cycle is an interrupt acknowledge which requires an interrupt vector on D0-D7 (except in the case of an autovectored interrupt, when U203 asserts AUTO*, which in turn asserts VPA* during the interrupt cycle, causing U201 to

fetch the autovectored interrupt service routine address for that level). Signal AUTO* is latched for the duration of the interrupt acknowledge cycle, to ensure that VPA* meets its required hold time from AS*. All other interrupt acknowledge signals are locked out during an autovectored interrupt acknowledge.

Device U203 also generates the signal MALL, which is high during an interrupt acknowledge cycle. MALL prevents any of the device select signals from becoming active during an interrupt acknowledge, and it prevents U202 from attempting to translate the address during an interrupt acknowledge by forcing MAS* low. At the same time, signal ALL (generated in circuit 201, as described above) is forced low to prevent U202 from interpreting the failure to translate the address as a fault.

Table 27 specifies the logical relationships between the input and output signals of device U203:

TABLE 27

```
MALL  = (!AS & FC0 & FC1 & FC2);

MIACK = !(A1 & !A2 & A3 & !AS & AUTO & FC0 & FC1 & FC2 & !INTIQR);

TIACK = !(!A1 & !A2 & A3 & !AS & AUTO & FC0 & FC1 & FC2 & !INTIQR);

DIACK = !(!A1 & A2 & !A3 & !AS & AUTO & FC0 & FC1 & FC2 & !INTIQR);

PIACK = !(A1 & !A2 & !A3 & !AS & AUTO & FC0 & FC1 & FC2 & !INTIQR);

VIACK = (!AS & AUTO & FC0 & FC1 & FC2 & INTIQR);

AUTO  = !(!AS & AUTO
          # A2 & A3 & !AS & FC0 & FC1 & FC2 & !INTIRQ
          # A1 & A2 & !AS & FC0 & FC1 & FC2 & !INTIRQ
          # !A1 & A3 & !AS & FC0 & FC1 & FC2 & !INTIRQ & !RICAV
          # A2 & !AS & FC0 & FC1 & FC2 & !INTIRQ & !SERAV).
```

CPU 1 includes two ROM units U302 and U502, preferably having 128K bytes total capacity. In Figure 6, units U302 and U502 are 27256 ROM units, although

other types of ROM units (including 2764/27128 and 27512 ROM units) may be substituted for such 27256 ROM units by configuring jumpers JP3. Since processor U201's data bus (D0-D15) has sixteen bits, ROM U502 contains the odd bits and ROM 302 contains the even bits. All reads from ROM will enable both ROM units U302 and U502.

DTACK* for a ROM access is generated by two flip-flops U705 in circuit 208 (shown in Figure 6). When signal ROM* is inactive (high), these flip-flops are set asynchronously. When ROM* goes low, a low level is clocked through these flip-flops in sequence. The DTACK* is thus delayed for a minimum of 100 ns from the falling edge of ROM*. DTACK* will be sampled on the next falling edge of the 10MHz clock, and data will be sampled on the next falling edge after that. Thus, there will be a 250 ns period from the time the ROM's are enabled to the clock edge at which data is sampled. Since a 10 ns setup time is required, the ROM's must have an access time from CS* of less than 240 ns.

CPU 1 preferably includes the I/O devices to be described next, with reference to Figure 7. Parallel interface and timer U206 (which preferably is a MC68230 integrated circuit) has the following features: a parallel port (Port A) for communicating with a printer interface; a parallel port (Port B) for communicating with a microdiskette drive controller subsystem; and a 24 bit programmable timer.

U206 automatically generates signal PDTACK* in response to an access from processor U201. Interrupts from device U201 may be from the timer on line IR4* or from the parallel ports on IR1*. A timer interrupt is serviced by driving TIACK* low during an interrupt acknowledge cycle. U206 will place the contents of its internal timer interrupt vector register TIVR on lines D0-D7, A port interrupt is serviced by driving PIACK*

low during an interrupt acknowledge cycle. U206 responds with the contents of its internal port interrupt vector register. The last two bits of this register indicate the source of the interrupt. Such an interrupt can occur as a result of an event on any of the handshake lines H1 through H4.

The following lines from U206 are used to implement the parallel printer interface: PA0-PA7, which are the outgoing data lines, and are buffered by buffer U015 to generate DATA1-DATA8 to the printer (Port A is configured as output only, and U015 is always enabled); PC4 (or "PAPER"), which goes high if the paper is out of paper; PC2 (or "BUSY IN"), which goes low when the printer is ready to accept data (the input from the printer is buffered by buffer U16) and may be read by processor U201 through Port C of U206; PC1 (or "SEL IN"), which goes high when the printer is on line (the input from the printer is buffered by buffer U16) and may be read by processor U201 through Port C of U206; PC0 (or "FAULT*), which goes low when the printer has experienced a fault condition (the input from the printer is buffered by buffer U16) and may be read by processor U201 through Port C of U206; H2 (or DSTB*), which goes low to indicate DATA1-DATA8 is valid (typically, U206 will be programmed to configure H2 as an output, and H2 is generated explicitly under program control); and H1 (or "ACK*"), which goes low to indicate that the printer has received data, and may be read through U206's internal storage register. Buffers U15, U16 are preferably 74LS244 integrated circuits. The printer reset, PRST*, may be set via DUART 68681.

The following lines from U206 are used to implement the interface to the microdiskette drive controller ("MDC") subsystem: PB0-PB7, a bidirectional data bus, which is buffered by buffer U205 (preferably a 74LS245

integrated circuit), which buffer is controlled by the MDC subsystem; H4, which is a handshaking signal for data transfers, and may be monitored by the MDC subsystem processor (typically, H4 is programmed to go high when a data byte is received or sent by U201. When this is sensed by MDC processor U403, the MDC processor will respond by accessing the data port, generating a pulse on line FW*. This completes the handshaking sequence, generating an interrupt to U201.); and H3 (FW*), which is the above-mentioned handshaking signal for data transfers.

U206 also includes a 24 bit programmable timer, and may optionally be clocked by a five bit prescaler. The timer operates as a down counter which is loaded from a preload register programmed by U201.

CPU 1 preferably includes an eight position dip switch for configuration, such as switch U102 shown coupled with octal buffer U101 in Fig. 6. The state of the switches is enabled onto D8-D15 whenever U206 is accessed.

The MDC subsystem comprises processor U403 (preferably a 6809E integrated circuit running at 2MHz), 8k x 8 RAM unit U404, 8k x 8 ROM unit U405, 765A floppy disk controller U406, and circuitry for clock generation, address decoding, and interfacing to the microdiskette drive. All communication between the MDC subsystem and processor U201 is through circuit U206.

The E and Q clocks for processor U403 are generated by circuit 220, which includes two flip-flops U607 driven by an 8 MHz clock derived form the 16 MHz system clock using flip-flop U606.

Address decoding and control signal generation are implemented by programmable logic device U506 and decoder U407. Device U506 generates the following signals for the MDC subsystem: FDCRD*, which is the FDC

read strobe (and is simply FRW qualified by FE); FDCWR*, which is the FDC write strobe (and is qualified by both E and Q to increase data setup times); FW*, which is the MDC subsystem strobe, and goes active whenever the data port to parallel interface U206 is accessed to generate an interrupt to processor U201; CSEN*, which is the chip select enable, and is used to separate two areas of the MDC subsystem memory map into two sections, based on FA13 (in each of these two areas, CSEN* will always be active when FA15 is high), and to enable decoder U407; TC, which is the terminal count, and is set high by U403 to signal the end of a data transfer; MOTORON*, which goes low to actuate the microdiskette drive motor; and EJECT*, which causes the diskette to be ejected when low. TC and MOTORON* are implemented as latches, whose operation is explained by the state transition tables set forth as Table 28:

TABLE 28

(FDCU*, FA1, FA0)

		000	001	011	010	110	111	101	100
TC	0	0*	0*	1*	0*	0*	0*	0*	0*
	1	1*	1*	1*	0	1*	1*	1*	1*

(FDCU*, FA2, FA0)

		000	001	011	010	110	111	101	100
MOTORON*	0	0*	0*	1	0*	0*	0*	0*	0*
	1	1*	1*	1*	0	1*	1*	1*	1*

* = Stable Statement

Table 29 specifies the logical relationships between the input and output signals of device U506:

TABLE 29

```
FRAR    =   !(F3 & FR_W);
FRAW    =   !(FE & FQ & !FR_W);
FW      =   !(!FQ13 & !FA14 & !FA15 & FE & FQ);
CSEN    =   !(!FA13 & FE # FA15 & FE);
FDCU    =   !(FA13 & FA14 & !FA15 & FE & FQ);
enable FD7  =  (FA13 & !FA14 & !FA15 & FE & FR_W);
FD7     =   !(!H4);
TC      =   !(!FA1 & !TC # !FA0 & FA1 & !FDCU # FDCU & !TC);
MOTORON =   !(!FA2 & !MOTORON # !FA0 & FA2 & !FDCU # FDCU & !MOTORON);
EJECT   =   !(MOTORON & TC).
```

The floppy disk controller (FDC) circuit U406 (preferably a 765A integrated circuit) provides control functions and data formatting for the microdiskette drive. Interface circuit U505 (preferably a 9229T integrated circuit) is coupled to U406, and provides data separation and write precompensation. The control lines to the microdiskette are buffered by open-collector devices U012, U304, and U110. Received lines are pulled to VCC with 2.7K resistors.

U406 interfaces to the following control lines: WPROT*, which senses a write protected disk; TRK 0*, which senses track 0 in the seek mode, and in read/write mode (indicated by a low on RW*/SEEK) is forced low to prevent U406 from detecting a fault condition; RDY*, which indicates the microdiskette is ready to receive dat; IDX*, which is the index to the beginning of a disk track; SIDE SEL*, which selects one side of a two-sided diskette; WRITE GATE*, which enables write data to the

diskette; DIR*, which sets the seek direction; and STEP*, which is forced high when the FDC is in its read/write mode. In addition to these lines, the following lines (coupled to buffer U012 or U110) are included in the microdiskette interface: RDATA*, which is the raw data read from the diskette drive; WDATA*, which is the MFM encoded data to the diskette drive; DS3*, DS1*, and DS0*, which are drive selects; DEJECT*, which when low, causes the drive to eject the diskette; and DMOTON*, which is allows U403 to turn off the drive motor.

Data separation and write precompensation are performed by interface circuit U505 (preferably a 9229T integrated circuit). U505 also provides clock signals for U406. Clock signal CLKOUT (the write clock to U406) has frequency 1 MHz, and clock signal HLT/CLK (the master clock to U406) has frequency 8 MHz.

FDCSEL and MINI are pulled up, and MFM driven high to configure the microdiskette controller and interface for double density MFM. Write compensation may be set to any value from 0 to 625 ns in 125 ns increments by configuring JP5.

CPU 1 includes an initiator for a Small Computer System Interface (SCSI). SCSI is a standard interface designed to facilitate integration of systems and peripherals, such as disk drives, from many manufacturers. CPU 1's SCSI interface includes SCSI controller U208 (preferably an NCR 5386 SCSI controller chip), which supports arbitration and reselection, and handles all low level protocol and timing. In CPU 1, circuit U208 is configured as an initiator with ID = 7. An initiator establishes a connection with a target to exchange data. Once the connection is established, through an arbitration and selection process, the target controls the entire transaction, requesting commands and

data, and returning status in the form of messages. The target may also disconnect from the initiator, and reconnect at a later time. Table 30 lists the registers of U208 (in an embodiment in which is an NCR 5386 circuit):

TABLE 30

Address	Register Type	Register description
$FFB000	R/W	Data Register
$FFB002	R/W	Command Register
$FFB004	R/W	Control Register
$FFB006	R/W	Destination ID
$FFB008	R	Auxiliary Status
$FFB00A	R	ID Register
$FFB00C	R	Interrupt Register
$FFB00E	R	Source ID
$FFB012	R	Diagnostic Status
$FFB018	R/W	Transfer Counter (MSB)
$FFB01A	R/W	Transfer Counter (2nd byte)
$FFB01C	R/W	Transfer Counter (LSB)
$FFB01E	R/W	Reserved
$FFB020	R/W	Virtual Port

Access to the SCSI interface is controlled by programmable logic device U011. The main function of U011 (which is preferably a PLS 153A integrated circuit) is as a controller for "DMA-like" access to U208 (where DMA denotes a conventional direct memory access technique). The DMA-like access technique of the invention is implemented in an asynchronous state machine described by the state transition diagram shown in Table 31:

TABLE 31

(VPORT*, DREQ)

(DACK*, DLYREQ1, DLYREQ2)	10	11	01	00
000	111	101	001	011
001	101	101	001*	001*
011	100	100	100	100
010	110	100	100	110
110	110*	100	100	110*
111	100	100	100	100
101	100	100	000	000
100	110	100*	000	010

* = Stable State.
VPORT* = !SCSI & A5

With reference to Table 31, the SCSI controller (U208) drives DREQ high to request DMA service. Two state variables, DLYREQ1 and DLYREQ2, delay the DMA acknowledge signal (DACK*) until the system processor (U201) accesses the virtual port, VPORT. This signals U208 that the transfer is underway. U208 then brings DREQ low in response to DACK*, and at the same time, U011 generates SDTACK* (the SCSI data transfer acknowledge signal) which informs the system processor (U201) that the transfer is complete. When VPORT* goes high, DACK* and SDTACK* go inactive. Signal SDTACK* is generated as a result of DMA acknowledge signal DACK* from U208. If for any reason, DACK* is not asserted after processor U201 attempts to access virtual port VPORT, SDTACK* will remain inactive, and processor U201 will halt execution until the bus times out, generating a bus error.

It is possible to transfer blocks of data through the virtual port using a tight loop of only two instructions. Processor U201 (in an embodiment in which U201 is a 68010 integrated circuit) has an internal

cache from which such loops may be executed without accessing main memory. We have recognized that when data transfers are implemented in this way, CPU 1 can maintain an extremely high data transfer rate in excess of 500,000 bytes per second. Such data transfer rate is greatly in excess of the data rate achievable using a conventional DMA technique.

Signal SCSI* goes low to access either U208 or the virtual port VPORT. Signal A5 determines which is selected (SCSI controller U208 is selected if A5 is "zero"; virtual port VPORT is selected if A5 is "one"). SDTACK* goes low when either of these devices is accessed.

Table 32 specifies the logical relationships between the input and output signals of device U011:

TABLE 32

```
SCS     =   !(!A5 & !SCSI);

DIST    =   !(E & RW & !VMA);

RIWR    =   !(E & !RW);

RIRD    =   !(E & RW);

DLYREQ1 =   !(DLYREQ2 # DREQ);

DLYREQ2 =   !(DACK # DLYREQ1);

SDTACK  =   !(!DACK & !PUDS # !A5 & !PUDS & !SCSI);

DACK    =   !(A5 & !DLRREQ1 & !SCSI);
```

The SCSI bus is terminated with the open collector devices shown connected to the output terminals of U208, including Hex Schmitt-trigger inverter U018, Texas Instruments 74LS642-1 circuit U017, inverter U702, and buffer circuits U114, U115, and U304. The SCSI bus comprises the following signals: SB0-SB7, the data bus which transmits all data, status, and messages (and

which is also used during arbitration by the initiator to identify itself, and during selection to identify the initiator and the target devices); SBP, which indicates parity (when the parity option is enabled, this line implements odd parity over SB0-SB7); BSY, which is asserted by the target to indicate that the SCSI bus is being used (during arbitration, the initiator must also assert BSY to indicate to the other devices on the SCSI bus that arbitration is occurring); SEL, which is driven low by the initiator to select a target (since SCSI allows a target to reselect an initiator following disconnection, the initiator must also monitor this line; C/D, which is driven low by the target to indicate that the bus contains control information instead of data; I/O, which is driven low by the target to indicate data input to the initiator, and is also used to distinguish between selection and reselection phases; MSG, which is driven low by the target during the message phase; REQ, which is driven low by the target to request a data transfer (the initiator responds by driving ACK low); ACK, which is driven low by the initiator to acknowledge a data transfer initiated by REQ; ATN, which is driven low by the initiator to indicate an ATTENTION condition; and RST, which is driven low by U201 to reset devices on the SCSI bus.

Read and write strobes for the SCSI controller are generated by random logic according to the following equations: !WR* = (!SCS* # !PUDS* # !R/W*) & (!DACK* # !PUDS* # !R/W*); !RD* = (!SCS* # !PUDS* # !R/W) & (!DACK* # !PUDS* # !R/W).

The first term in each equation represents an access of the controller (U208) itself, while the second term represents an access through the virtual port VPORT.

The SCSI controller (U208) may interrupt U201 by asserting SCSIIRQ. This request passes through a NAND circuit, gated by SCSIENB. Processor U201 may disable the SCSI interrupt by resetting SCSIENB, but at the same time, processor U201 may monitor SCSIIRQ via DUART 207.

CPU 1 includes dual universal asynchronous receiver/transmitter (DUART) U207 (preferably a 68681 integrated circuit), for serial communication. U207 provides two RS-232C serial ports (Ports A and B) for system console and integration, respectively. The clock for U207 is derived form the clock output of SC11004 single chip modem U707. The clock output of U707 has frequency equal to 7.3728 MHz, which is divided by two (by circuit U600) to generate the 3.6864 MHz clock required by U207.

RS-232C standard drivers, U014 and U011, and receivers U013 and U109, provide the interface U207 between and the serial lines. General purpose I/O lines in U207 are exploited for other purposes, including: SCRST* (SCSI reset); SCSIENB (SCSI interrupt enable); SCSIIRQ (SCSI interrupt request); DREQ* (SCSI controller DMA request); and PRST* (parallel printer reset).

Modem U707, which preferably is a Sierra Semiconductor SC11004 single chip modem, and modem controller U303, which preferably is a Sierra Semiconductor SC11007 parallel interface modem controller, are provided for facilitating remote diagnostics. Connected to U707 and U303 is a data access arrangement comprising a ring detector, isolation transformer, line seizure relay, and protection circuitry.

For accurate operation, external parallel resonant crystal Y4 (having 20 pF load), is coupled to modem U707, with a 27 pF capacitor between XTAL1 and ground and a 47 pF capacitor between XTAL2 and ground.

The interface to U303 is preferably identical to an 8259 UART.

CPU 1 includes real time clock U604 (preferably an Intersil 7170 circuit) for system timekeeping, including date and time, and periodic interrupts. Clock U604 interrupts U201 every 10ms with an autovectored, level four interrupt. The operating system for U201 may use this interrupt to update the time and date variables and to provide context switching. Clock U604 requires a 32.768 kHz crystal, Y3, whose frequency may be adjusted with variable capacitor C17. Since system time is kept from this device, and since U201's operating system will typically depend on accurate time for many of its operations, the frequency of clock U604 must be accurately adjusted (by programming U604 for one second interrupts, and adjusting C17 until the period is exactly one second).

U604 is a VPA device requiring special read and write strobes, RTWR* and RTRD*, which are generated by U011.

CPU 1 and PCI units 3 are controlled by a software operating system, so that the function of each PCI unit 3 may be independently defined by software downloaded from CPU 1. It is contemplated that the operating system may support a variety of voice and data processing application programs, including a voice mail application program. The operating system will control all input/output operations for the system, as well as the telephone interfacing operations performed by the PCI units 3. In a preferred embodiment, system processor U201 of CPU 1 is programmed to operate under an operating system which performs most (or all) of the functions of conventional operating systems (for example, the conventional VMS operating system available

from Digital Equipment Corporation), but which also performs the inventive functions to be described below.

An important advantage of the inventive system is that its hardware architecture (described above), together with its operating system, allows processing of digitized voice information together with other digital data. In the inventive hardware/software system, digitized voice and other data coexist in a single file structure. The file records may consist of fields containing numeric data information as well as fields containing recorded, digitized, voice information.

The inventive system stores digitized voice information in data records for playback in analog form (such as for transmission over a telephone line to a voice mail user). The inventive system also stores digitized voice information, together with DTMF data and other digital data, in data records for a wide variety of processing operations (such as compilation into customized voice or printed reports). The inventive system is capable of exchanging such data records with a mainframe computer for updating files or retrieval of data base information needed for a variety of voice information processing applications.

In a preferred embodiment, the inventive operating system performs many of the functions of the above-mentioned VMS operating system, including creation and maintenance of files, display or printing of file contents, configuration and maintenance of software applications, monitoring and control of system processes, and performance of basic system diagnostics. Conventional software algorithms may be used to perform such operations, with the exception of file creation (to be discussed below). Additionally, it is contemplated that an ordinarily skilled computer programmer, given the description provided in this Specification, will be

able readily to modify a conventional operating system so that the modified operating system may be used to record vocal phrases to be manipulated by a variety of applications programs.

Because the inventive system is designed to be particularly useful for supporting voice processing applications, it is important that the system open files as rapidly as possible. For example, in many applications (such as voice mail), the user will record voice information in response to a system-generated vocal prompt. If the system requires a long time to open a file for receiving the user-supplied voice message, users will experience an annoying delay between their response to the prompt, and an indication by the system that the system is ready to accept their message.

The operating system of a preferred embodiment of the invention allows rapid file creation by using a novel data construct known as a "pseudo-file", to defer many of the disk access steps required for file creation until after the user commences to supply file contents (for example, voice information) to the system. To create a new file, the inventive operating system will perform the following operations: (a) create a pseudo-file in cache memory (i.e., allocate an unused disk cache for file information, and write the file header into the cache memory); then (b) write the data buffer (the file data) onto the disk using file information in the cache memory; and then (c) at the close of the file, search for an empty directory entry and fill with the new filename, allocate disk space for file information, write to the disk the directory entry and the file information (including the file header) which had been kept in the cache memory. Preferably, step (c) will also include the final operation of making the file id invalid.

In contrast, conventional operating systems create a file in the following manner: (a) search for an empty directory entry and fill with the new filename, allocate disk space for file information, and write to the disk the directory entry and file information; then (b) open the file by searching for a directory entry that matches the filename, and read the file information into a disk cache; then (c) write the data buffer onto the disk using information in the disk cache; and then (d) at the close of the file, write the updated file information from the disk cache to the disk.

It will be appreciated that "write" step (b) of the inventive method corresponds to "write" step (c) of the conventional method. The inventive method defers several disk accesses to its final step (step (c), which follows the inventive "write" step), which disk accesses occur during steps (a) and (b) of the conventional method (i.e., before the "write" step of the conventional method).

The operating system of a preferred embodiment of the invention includes a "telephone command processor" algorithm (TCP) which allows a system user to modify the menu presented (vocally) to persons initiating telephone contact with one of the PCI units 3. TCP is a system-wide application access and menu manager, which generates "top level" vocal menus, each of which menus may offer callers the option to access either a submenu or an application program (such as voice mail). A preferred embodiment of TCP will be described with reference to the flow diagram of Figure 8.

Each port of each PCI unit 3 may be assigned a default port program that will automatically be executed when a caller telephones the port. The default port program may be a dedicated program which terminates a call, or allows the caller to enter the TCP "log on"

sequence, or executes a default application program. TCP will preferably prompt the caller to enter a user ID and password if no other default port program is assigned.

When a caller successfully logs on, TCP will check whether the caller's account is assigned an application. If so, TCP will execute such application program and hang up upon completion of the application program. A voice mail application program is a typical example (in which case, the voice mail application program will typically prompt the caller for a mailbox number and either record a message from the caller or allow the caller to hear recorded messages in the caller's mailbox). If no application is assigned for the caller's account, TCP will cause the PCI unit to present (orally) a TCP menu to the caller.

A typical TCP menu might include the following oral phrases: "To enter voice mail, press [1]. For additional options, press [8]. To end this call, press [9]." If the caller responds by pressing key [8] (thereby sending the PCI unit a recognizable DTMF signal), TCP might orally present a submenu such as the following to the caller: "For time and date, press [1]. To change your password, press [2]. To play [a voice game], press [3]. To exit this menu, press [9]." In this example, processor U201 would be programmed to execute a voice game application program in response to entry of the DTMF command "[3]".

It is contemplated that processor U201's operating system will offer a user the option of revising the TCP menus and submenus presented to callers. Preferably, the operating system allows the user to define TCP menu key assignments, to record new oral TCP menu phrases and create a voice file for each user-recorded menu phrase (which voice file contains a digitized version of the recorded menu phrase), and to copy such voice files to

the operating system directory so that TCP can access each such voice file.

For example, suppose the user wishes to offer a caller the option of executing a customized application program known as "Sales Administrator". Such an application program might prompt a salesman caller to enter the amount and date of each sale made in the last week (by entering DTMF tones). The program might then process the tones entered by a group of salesmen to compile a voice report listing cumulative sales data. Such voice report might consist of concatenated digitized voice phrases, some of which represent the result of mathematical processing of a set of numerical signals entered by the salesmen (i.e., the program might translate all DTMF "sales price" signals into numbers, and then add the numbers, and then translate the number representing the sum into a voice file containing a prerecorded digitized vocal representation of the sum). The application program might also prompt callers to exercise their option to hear the most recently updated version of such voice report.

Assuming such an application program has been coded, and entered as an executable application program that may be executed by the operating system of the invention, the user might modify the TCP submenu set forth above to add to it a fifth phrase as follows: "To execute Sales Administrator, press [4]." In order to so modify the submenu, the user would need to record the indicated phrase, and store it in a voice file that may be accessed by TCP.

The operating system of a preferred embodiment of the invention also includes an algorithm allowing users to create files known as "tagged messages". Each tagged message may include: (a) data (such as numerical data, or a digitized voice signal, or both), (b) a field

identifying the file as a "tagged message", (c) a field specifying the file name of another file (which other file contains a digitized vocal prompt message), and (d) a field specifying an application program to be executed in response to a command (i.e., a DTMF command supplied by the recipient of the prompt identified in field (c) above).

An important example of the utility of tagged messages will be described with reference to a voice mail application program. When a voice mail recipient encounters a tagged message in reviewing the voice messages in his or her mail box, the inventive system will play back the digitized voice content of the tagged message just as it would play back an untagged message. However, after the tagged message as been played, and the voice mail program is prompting the recipient for the usual optional actions to be taken in response to the message (i.e., the options of listening to the message again, deleting it, storing it, and so on), an additional prompt will be presented. This additional prompt is the one identified in field (c) of the tagged message. If the recipient then presses the telephone key indicated by the additional prompt, the application program identified in field (d) of the tagged message will be executed (as a sub-process of the on-going voice mail operation).

For example, the voice content of the tagged message (the contents of field (a)) might set forth a meeting agenda, and the prompt identified in field (c) of the tagged message might ask the recipient to press key [2] to initiate a "Meeting Scheduler" application program. Such "Meeting Scheduler" application program might allow the recipient to specify his or her availability for attending a meeting. Upon completion of execution of the "Meeting Scheduler" application, the

recipient would return to the voice mail application (so that he or she could continue to review messages in his or her mail box).

The "pseudo-file", "tagged messages" and TCP data constructs and methods described above will preferably be embodied in an operating system for the system described above with reference to Figures 1-7. However, the inventors specifically contemplate that such data constructs and methods may be embodied in other voice processing systems.

The invention also comprises improved methods for automated processing of incoming and outgoing telephone calls. These methods (to be discussed below with reference to Figures 9-14) will preferably be implemented in software. Such software will preferably be downloaded from processor U201 to port processor U87 for controlling a port of PCI unit 3 in operation of the system of Figs 1-7. However, the inventors specifically contemplate that the methods may be embodied in other voice processing hardware/software systems.

A preferred embodiment of one inventive call processing method will next be described with reference to Figs. 9-13. The Fig. 9 flow chart contemplates that a caller has completed a call to a port of PCI unit 3, that an automatic call distribution application program has commenced execution or that an outdial application has commenced execution. Fig. 9 further contemplates that the automatic call distribution application program has accepted from the caller a particular telephone number to be dialed, and has dialed that telephone number (in preparation for transferring the caller to the requested line) or that the outdial application program has dialed the target telephone number (in preparation for delivery of a message to the target party). The first block in the Fig. 9 flow chart

indicates that, at this point, PCI unit 3 will disable the voice sampling algorithm until the first positive edge (a transition from an energy line state to a no energy line state) thus reducing the chance of pre-connect line noise being interpreted as voice and triggering a premature voice detect response. PCI unit 3 will then monitor the status of the distant end of the dialed line for rising or falling edges (indicative, for example, of "ringing" or "busy" conditions).

If no positive edge is detected on the dialed line after 15 seconds have elapsed, PCI unit 3 will reconnect with the caller, and send CPU 1 a message indicating that the desired line is dead (or send the caller a voice message indicating that the desired line is dead).

If a positive edge is detected within 15 seconds after the Fig. 9 algorithm has commenced, the PCI unit will determine whether the edge detected was more than 600 ms after the Fig. 9 algorithm commenced. If so, the PCI unit will perform the "Wait for answer" operation described by the Fig. 13 flow chart.

Fig. 13 indicates that the PCI unit will report to CPU 1 that the called party has answered, if no additional edge (i.e., no second edge in addition to the first edge already detected) is detected within 4.3 seconds. If a second edge is detected within such 4.3 second period, but the overall RNA timer (which started at the beginning of the call progress algorithm) has expired, then the PCI unit will generate a report for CPU 1 (or a voice report for the caller) indicating that the called party's line did ring, but that the called party did not answer (i.e., that a "RNA", or "ring, but no answer" condition exists). If a second edge is detected within the 4.3 second period, and no third edge is detected within a 2.3 second period, the PCI unit will report to CPU 1 that the called party has answered.

PCI unit 3 continues monitoring the line for edge pairs until either an individual edge timeout occurs, the overall RNA timeout occurs, or voice detect occurs.

Returning to Figure 9, if the first edge was detected within a 600 ms period after the Fig. 9 algorithm commenced, PCI unit 3 will determine if a second edge is detected within an additional 575 msec period. If not, the PCI unit will return to position "C" in the Figure 9 flow chart, and continue to monitor the distant end for a signal edge. If so, the PCI unit will determine whether it should report a "busy" condition, a "reorder" condition, or should return to position "C" of the Fig. 9 flow chart.

PCI unit 3 will report a busy condition if the indicated conditions are met for progress from position J of Fig. 9, through positions I, L, and M, to position N. PCI unit 3 will report a "reorder" condition if the indicated conditions are met for progress from position J of Fig. 9, through positions I, L, M, O, and P, to position Q. Otherwise, PCI unit 3 will return to position C on the Fig. 9 flow chart.

The edge waiting algorithm shown in Fig. 10 will be performed at each of positions B, F, H, I, and K of Fig. 9. At position W1 of the Fig. 10 algorithm, PCI unit 3 will perform the Fig. 11 algorithm to get a 20 ms sample representing the status of distant end. The first step (Y1) of the Fig. 11 algorithm is described in Figure 12.

A source code listing (in Assembly language), of the preferred implementation of the call progress method described above with reference to Figs. 9-13, is set forth in this Specification below as Table 33.

A preferred embodiment of another inventive incoming call processing method, for accomplishing a call transfer, will next be described with reference to Fig. 14. The inventive call transfer method contemplates

that a caller has completed a call to a port of PCI unit 3, that an automatic call distribution application program has commenced execution. The automatic call distribution application program first plays a recorded voice message to the caller, which prompts the caller to enter DTMF signals indicating a desired telephone number. The example of the inventive method shown in Figure 14 contemplates that the inventive system is coupled to a conventional PBX system, and that the "desired number" a multi-digit extension number (XXX) on the PBX network.

The next step in the inventive call transfer method is for PCI unit 3 to execute a "hook flash" ("HF"), or PBX soft hold operation, to place the caller on hold. PCI unit 3 then connects with the PBX to dial the desired number and hangs up ("drops the line", as indicated in Fig. 14), connecting the caller with the desired called party's line. By performing this operation, PCI unit 3 is said to have executed a "blind transfer". The caller will hear the called party's phone ringing at this stage (after execution of the blind transfer), if the called party's line is idle.

The next steps of the Fig. 14 method rely on the feature of most conventional PBX systems which automatically reconnects a master station (PCI unit 3, in the present case) to a caller station when the master station goes off hook after the master station executes a blind transfer to a third station (a "called party" station) on the PBX network, if the called station's called party's line is busy or in error (and the caller station does not itself hang up).

In the Fig. 14 method, after PCI unit 3 drops the line, it goes off hook and listens for a call progress tone. If no dial tone is detected after a user-defined period of time (which may be a fraction of a second, or

as long as 2000 ms) then it is assumed that the called party's line was busy and the PBX has reconnected the PCI unit with the caller station. In this case, branch Z3 of the Fig. 14 method is executed, and the PCI unit plays a recorded message to the caller indicating that the called party's line is busy, or unavailable. Such recorded message may offer the caller other menu options, such as the option to hold (in which case the PCI unit will redial the called party's line after waiting for a predetermined period).

After PCI unit 3 drops the line, goes off hook and detects a dial tone, PCI unit 3 will wait for a user-defined period of time (for example, up to seven seconds) after detecting the dial tone in order to verify the dial tone identification. If a line break occurs during this period, and the dial tone is not verified, branch Z3 of the Figure 14 method is executed. If the dial tone is verified, branch Z1 or branch Z2 of the Fig. 14 method will be executed.

Either of three conditions will give rise to detection of a dial tone under these circumstances: either the calling party is connected with the called party (their conversation is in progress); or the called party's line is still ringing; or the calling party decided to abandon the call attempt, and had hung up his or her phone. When the PCI unit detects a verified dial tone under the described circtumstances, it will stay off-hook to keep the line/station busy for a selected period (preferably 20 seconds) so that the PBX will not force any additional calls in. At the end of this period, the PCI will execute a hook flash and attempt to answer the called party's line using the PBX's directed call pick up feature. If a "reorder" tone or "error" tone is heard following such directed call pickup attempt, the PCI unit goes on hook (under branch Z1 of

the Fig. 14 method), as it is assumed that either the called party answered the line before the directed call pick up attempt, or that the calling party has hung up (although there are other possibilities). If no reorder or error tone is heard following such directed call pickup attempt, it is assumed that the directed call pick up was successful, and that the PCI unit has been reconnected with calling party. In this event, branch Z2 of the Fig. 14 method is executed, and the PCI unit plays a recorded message to the caller indicating that the called party is unavailable. Such recorded message may offer the caller other menu options, such as the option to dial another extension.

An important advantage of the inventive method described with reference to Figure 14 is that the called party is immediately and directly connected with the calling party when the called party goes off hook to service the call. In conventional call transfer systems of this class, there is a delay in connecting the calling and called parties, when performing transfers with ring-no-answer supervision, which may lead to the called party's initial greeting being completely lost in the transmission. Such a delay may cause the called party to become confused, since he picked up his phone, expressed a greeting, and did not receive a reply. The caller may also become confused at this point, when employing such a conventional system, since he will not hear the called party's initial greeting.

Among the other advantages of the Fig. 14 method is that if the called party's line rings upon the blind transfer, the calling party will continue to hear a ringing line until the PCI unit intercepts the calling party's line. During the ringing period, the calling party is less likely to become discouraged and hang up than in the case that no ringing tone is audible (as is

the case in conventional automated call transfer systems, such as conventional systems operating "behind" a PBX (not on the CO or a trunk side) that operate, electronically, like a single-line 2500-type telephone). Another advantage of the Fig. 14 method is that if the called party does answer the line, the calling and called party will be completely unaware that the PCI unit attempts a directed call pick up during their conversation. In contrast, in conventional automated call transfer systems in which the master station does not execute a blind transfer (and instead remains off-hook during the transfer process), when the called party answers its line, the called party will be aware that the master station is off-hook and connected with the called party (due to noise or intentional signals introduced on the line by the master station). Finally, in the event that the called party's line is busy, the PCI unit of the invention is immediately available to assist the calling party.

A source code listing (in Assembly language), of the preferred implementation of the call transfer method described above with reference to Figure 14, is set forth below in this Specification as Table 34.

The above description is merely illustrative and explanatory of the present invention. Various changes in details of methods and apparatus described may be within the scope of the appended claims without departing from the spirit of the invention.

TABLE 33

```
Error
Address  Code      Sequence  Source Statement
                     2514  *
                     2515  *   CALL PROGRESS DETECTION
                     2516  *   Returns:
                     2517  *       A = $01 : ring answered
                     2518  *       A = $82 : silence
                     2519  *       A = $83 : busy
                     2520  *       A = $84 : reorder tone
                     2521  *       A = $85 : ring no answer
                     2522  *       A = $86 : answer detected and message required
                     2523  *
                     2524  *
                     2525  *
                     2526  *   If R passed in A,monitor for ringback only-if A,also monitor for
                     2527  *   answer.  Timeout for call prog(EXTTMO or INTTMO) passed in in X.
DDE5   7D 0036       2528  RESULT  TST     HUNGUP  Have we already had a hangup interrupt?
DDE8   27 03         2529          BEQ     1$      No, then we can proceed with call progress
DDEA   86 01         2530          LDAA    #$01    Yes, then fake an answer
DDEC   39            2531          RTS
DDED   C6 01         2532  1$      LDAB    #1      Flag that we are doing call progress
DDEF   F7 D03E       2533          STAB    CPENBL
DDF2   C6 C0         2534          LDAB    #$C0    Set bit 6 of IRQENB to enable T1
DDF4   D7 AE         2535          STAB    IRQENB
DDF6   B7 D024       2536          STAA    MTRTYP  Save progrss detection type
DDF9   FF D01D       2537          STX     TMR_C1
DDFC   7F D025       2538          CLR     ENERGY  Reset call progress
DDFF   7F D026       2539          CLR     LSTPCK
DE02   7F D027       2540          CLR     NRGCNT
DE05   7F 004A       2541          CLR     EDGFLG  Start with no edge detected yet
DE08   7F D02C       2542          CLR     DTEKT   Clear the voice detection counter
DE0B   86 14         2543          LDAA    #20     Initialize 20 msec timer
DE0D   B7 D023       2544          STAA    THR_CA
DE10   96 38         2545          JSR     DODSI   Disable silent interrupts
DE13   96 38         2546          LDAA    VIARA
DE15   8A 02         2547          ORAA    #$02    Force zero/APT high (ready for speech)
DE17   84 FB         2548          ANDA    #$FB    CVSD direction = input
DE19   97 38         2549          STAA    VIARA
DE1B   97 A1         2550          STAA    CVSDIO
DE1D   86 FF         2551          LDAA    #$FF    Set clamp for recording
DE1F   87 1000       2552          STAA    VOLUME
DE22   86 05         2553          LDAA    #5      Set unknown state voice detect sensitivity
DE24   87 D033       2554          STAA    VOCSEN
                     2555  *
DE27   CE 3200       2556          LDX     #COMRAM Start of debug data
                                                   +$200
DE2A   FF D029       2557          STX     DEBUG
                     2558  *
```

TABLE 33

Error Address	Code	Sequence	Source Statement			
		2559	*	Wait for energy to begin, then time the energy to determine if		
		2560	*	ringback, busy, or reorder. If >750 msec, assume ringback.		
		2561	*	Otherwise check for noise or errors.		
DE2D	7F D02E	2562	WATNRG	CLR	ERRFLG	Indicate not in error checking
DE30	7F D030	2563		CLR	SAMFLG	Set the no samples flag
DE33	FE D01D	2564	WATNR2	LDX	TMR_C1	
DE36	26 11	2565		BNE	5$	Branch if not timed out
DE38	86 85	2566		LDAA	$85	Timed out with no answer
DE3A	20 0A	2567		BRA	4$	
DE3C	86 82	2568	1$	LDAA	#82	Dead line
DE3E	7D 004A	2569		TST	EDGFLG	
DE41	27 03	2570		BEQ	4$	If edge flag clear, report dead line
DE43	7E DEBE	2571		JMP	CPM1XT	Report answer detected
DE46	7E DEC9	2572	4$	JMP	CPMRXT	
DE49	7D 004A	2573	5$	TST	EDGFLG	If clear then 1st pass
DE4C	27 0F	2574		BEQ	2$	Allow 15 sec for timeout
DE4E	CE 1130	2575		LDX	#4400	Otherwise timeout according to best knowledge
DE51	BD DF2F	2576		JSR	GETEDG	
DE54	2A 03	2577		BPL	14$	No answer detected
DE56	7E DEBE	2578		JMP	CPM1XT	ANSWER detected!!
DE59	27 E1	2579	14$	BEQ	1$	Will assume answer
DE5B	20 21	2580		BRA	6$	Branch around code with 15 sec timeout
DE5D	CE 1130	2581	2$	LDX	#4400	first pass through, check first for ring
DE60	BD DF2F	2582		JSR	GETEDG	
DE63	2A 03	2583		BPL	18$	No answer detected
DE65	7E DEBE	2584		JMP	CPM1XT	ANSWER detected!!
DE68	26 14	2585	18$	BNE	6$	An edge was received
DE6A	7D D025	2586		TST	ENERGY	Was what we were timing silence?
DE6D	27 05	2587		BEQ	7$	Yes . . . allow more time
DE6F	7C 004A	2588		INC	EDGFLG	Force assumed answer
DE72	20 C8	2589		BRA	1$	
DE74	CE 32C8	2590	7$	LDX	#13000	Allow 13 seconds total for edge
DE77	BD DF2F	2591		JSR	GETEDG	
DE7A	2B 42	2592		BMI	CPM2XT	ANSWER detected!!
DE7C	27 BE	2593		BEQ	1$	No edge in 13 seconds total . . . dead line
DE7E	7D D025	2594	6$	TST	ENERGY	See if new state is silence
DE81	26 B0	2595		BNE	WATNR2	If not go time energy in progress
DE83	86 02	2596		LDAA	#$02	
DE85	C6 58	2597		LDAB	#88	
DE87	BD DFDE	2598		JSR	CMXAB	Compare X to 600 msecs
DE8A	26 03	2599		BNE	3$	Branch X>600
DE8C	7E DECD	2600		JMP	CKERR	<=600 msecs, so check for busy or reorder
		2601	*	If MTRTYP set do answer detect-otherwise return success code		
DE8F	B6 D024	2602	3$	LDAA	MTRTYP	Get the progress detection type
DE92	81 52	2603		CMPA	#'R	*Is* it ring only?
DE94	27 28	2604		BEQ	CPM1XT	Yes - exit with success now
DE96	B6 D032	2605		LDAA	VOCDET	Get user defined detection threshold

TABLE 33

```
Error
Address  Code      Sequence  Source Statement
DE99     B7 D034   2606      STAA   VOCWRK  Set working threshold
DE9C     86 05     2607      LDAA   #5      Set known state voice detect sensitivity
DE9E     B7 D033   2608      STAA   VOCSEN
                   2609  *
DEA1     CE 10CC   2610  ANSWER LDX   $4300  Maximum silence time
DEA4     BD DF3F   2611      JSR    GETEDG
DEA7     2B 15     2612      BMI    CPM1XT  Voice detected!
DEA9     27 13     2613      BEQ    CPM1XT  No edge, assume connected
DEAB     FE D01D   2614      LDX    THR_C1  Has the no answer monitor timer expired?
DEAE     26 04     2615      BNE    1       No . . . proceed with check
DEB0     86 85     2616      LDAA   #85     monitor timeout, no answer
DEB2     20 15     2617      BRA    CPMRXT  Return
DEB4     CE 08FC   2618  1$  LDX    #2300   Maximum energy time
DEB7     BD DF2F   2619      JSR    GETEDG
DEBA     2B 02     2620      BMI    CPM1XT  Voice detected!
DEBC     26 E3     2621      BNE    ANSWER  Go validate the new 'ring'
                   2622  *
                   2623  *               ANSWER DETECTED OR ASSUMED
                   2624  *
DEBE     86 01     2625  CPM1XT LDAA   #1      Assume answer with no message
DEC0     F6 D024   2626      LDAB   MTRTYP  Get progress detection type
DEC3     C1 4E     2627      CMPB   #'N     Check if message required
DEC5     26 02     2628      BNE    CPMRXT  No
DEC7     86 86     2629      LDAA   #$86    Answer with message required
                   2630  *
                   2631  *               End Call Progress Monitoring with status in A reg.
                   2632  *
DEC9     BD DDD6   2633  CPMRXT JSR    STOPT1  Disable T1 and flag "end of call progress"
DECC     39        2634      RTS
                   2635  *
                   2636  *       Ringback was not detected, but some other cadence or
                   2637  *       noise was.  Try to identify the cadence ... if it's noise
                   2638  *       go back and re-monitor.  Otherwise, identify the cadence
                   2639  *       and return the proper error code.  Cadence is determined
                   2640  *       over the next four seconds.  If any edge doesn't happen
                   2641  *       within a second, skip the probable noise and look for
                   2642  *       ringback again.
                   2643  *
DECD     86 01     2644  CKERR  LDAA   #$01
DECF     C6 2C     2645      LDAB   #$2C
DED1     BD DFDE   2646      JSR    CMXAB   Compare X to 300 msecs
DED4     27 06     2647      BEQ    10$     X<=300
DED6     B6 D032   2648      LDAA   VOCDET  X>300;
DED9     B7 D034   2649      STAA   VOCWRK  Set user defined voice detect threshold
DEDC     CD 0FA0   2650  10$ LDX    #4000   Check edges for 4 secs
DEDF     FF D01F   2651      STX    THR_C2
DEE2     7F D028   2652      CLR    EDGCNT  Clear edge count
```

TABLE 33

Error Address	Code	Sequence		Source Statement		
DEE5	CE 023F	2653		LDX	#575	
DEE8	BD DF2F	2654		JSR	GETEDG	
DEEB	2B D1	2655		BMI	CPM1XT	ANSWER detected!
DEED	26 03	2656		BNE	2$	
DEEF	7E DE33	2657		JMP	WATNR2	Can't be error tone
DEF2	7C D028	2658	2$	INC	EDGCNT	Count up an ege
DEF5	CE 02EE	2659		LDX	#750	
DEF8	BD DF2F	2660		JSR	GETEDG	
DEFB	2B C1	2661		BMI	CPM1XT	ANSWER detected!
DEFD	27 BF	2662		BEQ	CPM1XT	No ege detected, assume ANSWER
DEFF	7F D030	2663		CLR	SAMFLG	Set the no samples flag
DF02	7C D02E	2664		INC	ERRFLG	Indicate into error checking
DF05	FE D01F	2665		LDX	TMR_C2	Timed out
DF08	26 E8	2666		BNE	2$	No timeout, keep counting
DF0A	7C D028	2667		INC	EDGCNT	One more edge
DF0D	BD DDD6	2668		JSR	STOPT1	Disable T1 and flag "end of call progress"
DF10	B6 D028	2669		LDAA	EDGCNT	How many edges?
DF13	7F D028	2670		CLR	EDGCNT	Clear the edge counter for WATNRG
DF16	81 05	2671		CMPA	#5	Check to see if it is busy
DF18	23 12	2672		BLS	4$	Not likely to be a busy tone if less frequent
DF1A	81 0A	2673		CMPA	#10	Is it in this realm?
DF1C	22 03	2674		BHI	3$	No ... may be reorder
DF1E	86 83	2675		LDAA	#$83	Show busy
DF20	39	2676		RTS		
DF21	81 0D	2677	3$	CMPA	#13	See if in the realm of reorder
DF23	23 07	2678		BLS	4$	Not really, unknown signal
DF25	81 16	2679		CMPA	#22	
DF27	22 03	2680		BHI	4$	Far too frequent for even reorder to be
DF29	86 84	2681		LDAA	#$84	Reorder
DF2B	39	2682		RTS		
DF2C	7E DE2D	2683	4$	JMP	WATNRG	Return on false detection
		2684	*	Use EDGE to find the next edge in X msces. Return the		
		2685	*	number of msecs actually used in X		
DF2F	FF D021	2686	GETEDG	STX	TMR_C3	Use edge detection timer
DF32	7F 004B	2687		CLR	SAVEDX	Indicate no edge
DF35	FF D035	2688		STX	XTEMP	Save original value of X
DF38	FE D021	2689	1$	LDX	TMR_C3	Hvae we timed out?
DF3B	27 0C	2690		BEQ	2$	Yes
DF3D	8D 1A	2691		BSR	EDGE	See if this was an edge
DF3F	2B 17	2692		BMI	30$	ANSWER may be detected!
DF41	27 F5	2693		BEQ	1$	No edge found yet
DF43	7C 004B	2694		INC	SAVEDX	Indicate edge received
DF46	7C 004A	2695		INC	EDGFLG	Set flag indicating edge has been seen
DF49	B6 D035	2696	2$	LDAA	XTEMP	Get original timer count
DF4C	F6 D036	2697		LDAB	XTEMP+1	
DF4F	FE D021	2698		LDX	TMR_C3	Determine number of msecs used
DF52	BD DFCB	2699		JSR	SUB2	

TABLE 33

Error Address	Code	Sequence	Source Statement			
DF55	7D 0048	2700		TST	SAVEDX	Get edge flag
DF58	39	2701	30$	RTS		
		2702	*	Debounce the line's energy state. If it switches to		
		2703	*	a new energy level, return the new energy status		
		2704	*	in ENERGY and the edge condition, NE. If no edge		
		2705	*	is detected, return EQ with ENERGY set appropriately		
DF59	8D 4A	2706	EDGE	BSR	CKLINE	Check to see if it is energy
DF5B	2B 47	2707		BMI	30$	ANSWER detected!
		2708	*			
		2709	*	LDX	DEBUG	
		2710	*	CPX	#COMRAM+7FF	
		2711	*	BEQ	1$	
		2712	*	STAA	X	
		2713	*	INX		
		2714	*	STX	DEBUG	
		2715	*			
DF5D	16	2716	1$	TAB		Save current value
DF5E	B8 D026	2717		EORA	LSTPCK	HI:changed since last packet
DF61	97 4C	2718		STAA	SAVEDX	Calling routine needs XTEMP,SAVEDX
					+1	
DF63	F7 D026	2719		STAB	LSTPCK	Set up for next pass
DF66	F8 D025	2720		EORB	ENERGY	HI:changed from debounced value
DF69	17	2721		TBA		
DF6A	D4 4C	2722		ANDB	SAVEDX	HI:could be start of new state
					X1	
DF6C	43	2723		COMA		HI:not changed from debounce
DF6D	94 4C	2724		ANDA	SAVEDX	HI:gone back to valid state
					X1	
DF6F	27 05	2725		BEQ	2$	No
DF71	7F 0027	2726		CLR	NRGCNT	Yes, clear counter
DF74	20 2D	2727		BRA	20$	
DF76	5D	2728	2$	TSTB		Has the line condition gone to new state?
DF77	27 08	2729		BEQ	3$	No
DF79	B6 D02D	2730		LDAA	EDGDET	Debounce time (data 6 gives 7 packets)
DF7C	B7 D027	2731		STAA	NRGCNT	Set debounce timer
DF7F	20 22	2732		BRA	20$	
DF81	B6 D027	2733	3$	LDAA	NRGCNT	Get current timer
DF84	27 1D	2734		BEQ	20$	Nothing timing
DF86	7A D027	2735		DEC	NRGCNT	Decrement timer
DF89	26 18	2736		BNE	10$	No timeout here
DF8B	7D D02E	2737		TST	ERRFLG	Are we into error checking?
DF8E	26 05	2738		BNE	10$	Yes, skip sample flag tests
DF90	86 FF	2739		LDAA	#$FF	New state is energy, allow samples (11 windows)
DF92	B7 D030	2740		STAA	SAMFLG	
DF95	86 03	2741	10$	LDAA	#3	Set next edge to 60-80 ms.
DF97	B7 D02D	2742		STAA	EDGDET	
DF9A	B6 D026	2743		LDAA	LSTPCK	Get (current) packet

TABLE 33

```
Error
Address  Code     Sequence  Source Statement
DF9D     B7 D025  2744            STAA    ENERGY  Save new energy state
DFA0     86 01    2745            LDAA    #1      Return NE, edge detected
DFA2     39       2746            RTS
DFA3     4F       2747    20$     CLRA            No edge
DFA4     39       2748    30$     RTS
                  2749    *
                  2750    *       Check for energy detection
                  2751    *
DFA5     BD E741  2752    CKLINE  JSR     GSAMP   Get a sample
DFA8     27 0F    2753            BEQ     30$     Voice not detected
DFAA     7C D02C  2754            INC     DTEKT   Voice detected, increment the count
DFAD     B6 D02C  2755            LDAA    DTEKT
DFB0     B1 D034  2756            CMPA    VOCWRK  At the voice detection threshold?
DFB3     23 07    2757            BLS     50$     No, voice not detected
DFB5     86 FF    2758            LDAA    #$FF    Flag that voice detected! (ANSWER!!)
DFB7     20 11    2759            BRA     90$
DFB9     7F D02C  2760    30$     CLR     DTEKT   Voice not detected, reset detection count
                  2761    *
DFBC     B6 D023  2762    50$     LDAA    TMR_C4  Be sure to wait the 20 ms minimum
DFBF     26 E4    2763            BNE     CKLINE  Has not elapsed, try another sample
DFC1     86 14    2764            LDAA    #20     Reset the 20 ms counter
DFC3     B7 D023  2765            STAA    TMR_C4
DFC6     96 A0    2766            LDAA    DTMFCP  Load call progress
DFC8     84 01    2767            ANDA    #$01    Isolate bit for energy
DFCA     39       2768    90$     RTS
                  2769    *


                  3951    *       GSAMP - Gets a voice data sample and returns sample type
E741     7D D030  3952    GSAMP   TST     SAMFLG  Take a sample?
E744     26 03    3953            BNE     10$     Yes, maybe
E746     7E E7DE  3954            JMP     GSAM10  No, indicate voice not detected
E749     96 A0    3955    10$     LDAA    DTMFCP  Load call progress
E74B     84 01    3956            ANDA    #$01    Isolate bit for energy
E74D     26 03    3957            BNE     40$     Energy present, take a sample
E74F     7E E7DE  3958            JMP     GASM10  Indicate voice not detected
                  3959    *
                  3960    *       Note:  don't trash A, since we pass it along for later
                  3961    *
E752     CE 3100  3962    40$     LDX     #COMRAM Place samples at $100 to $140 (65 samples)
                                          +$100
E755     96 38    3963            LDAA    VIARA
E757     8A 02    3964            ORAA    #$02    Force zero/APT high (ready for speech)
E759     84 FB    3965            ANDA    #$FB    CVSD direction = input
E75B     88 08    3966            EORA    #$08                                    (+2/16)
E75D     97 A1    3967            STAA    CLOCK                                   (+4/20)
                  3968    *
```

TABLE 33

Error Address	Code	Sequence		Source Statement			
		3969	*	Receive sample data bytes			
		3970	*				
E75F	96 A1	3971	50$	LDAA	CVSDIO		(-3/ 3)
E761	84 01	3972		ANDA	#$01	Mask in data bit	(-2/ 5)
E763	A7 00	3973		STAA	X	Save this value	(-6/11)
E765	96 A1	3974		LDAA	CLOCK	Clock CVSD	(-3/14)
E767	88 08	3975		EORA	#8		(-2/16)
E769	97 A1	3976		STAA	CLOCK		(-4/20)
		3977	*				
E76B	08	3978		INX		Bump the buffer pointer	(+4/ 4)
E76C	8C 313D	3979		CPX	#COMRAM +$13D	Terminate loop?	(+3/ 7)
E76F	27 08	3980		BEQ	60$	Yes	(+4/11)
E771	96 A1	3981		LDAA	CLOCK	No, clock CVSD	(+3/14)
E773	88 08	3982		EORA	#8		(+2/16)
E775	97 A1	3983		STAA	CLOCK		(+4/20)
E777	20 E6	3984		BRA	50$		(+4/24)
		3985	*				
E779	96 A0	3986	60$	LDAA	DTMFCP	Load call progress	
E77B	84 01	3987		ANDA	#$01	Isolate bit for energy	
E77D	27 5F	3988		BEQ	GSAM10	If currently no energy, then no voice!	
E77F	CE 30FF	3989		LDX	#COMRAM Index Samples +$FF		
E782	C6 FF	3990		LDAB	#$FF	Set B register to -1	
E784	08	3991	70$	INX		Bump buffer pointer	
E785	8C 313D	3992		CPX	#COMRAM +$13D	Terminate loop?	
E788	27 08	3993		BEQ	80$	Yes	
E78A	A6 00	3994		LDAA	X	Get next delta	
E78C	26 F6	3995		BNE	70$	Leave 1's alone	
E78E	E7 00	3996		STAB	X	Convert 0's to -1	
E790	20 F2	3997		BRA	70$		
		3998	*				
E792	CE 30FF	3999	80$	LDX	#COMRAM +$FF	Index samples at $2100 to $2140 (65 samples)	
E795	5F	4000		CLRB		Start at relative 0 point	
E796	7F D02F	4001		CLR	MINMAX	Indicate no critical points encountered	
E799	7F D02B	4002		CLR	DSTATE	Start at delta state 0; undertermined	
E79C	08	4003	82$	INX		Bump buffer pointer	
E79D	8C 313D	4004		CPX	#COMRAM +$13D	Terminate loop?	
E7A0	27 34	4005		BEQ	100$	Yes	
E7A2	EB 00	4006		ADDB	X	Add next delta	
E7A4	2B 19	4007		BMI	90$	Into negative delta	
E7A6	C1 01	4008		CMPB	#1	Have we moved up 2 yet?	
E7A8	23 F2	4009		BLS	82$	No	
E7AA	7D D02B	4010		TST	DSTATE	What state are we in?	

TABLE 33

Error Address	Code	Sequence		Source Statement		
E7AD	2B 05	4011		BMI	84$	We are neg and detected pos delta
E7AF	27 03	4012		BEQ	84$	Undertermined, bump critical count
E7B1	5F	4013		CLRB		Reset to relative zero point;
E7B2	20 E8	4014		BRA	82$	we are pos and detected pos delta
E7B4	7C D02F	4015	84$	INC	MINMAX	Bump critical count
E7B7	5F	4016		CLRB		Reset to relative zero point
E7B8	86 01	4017		LDAA	#1	
E7BA	B7 D02B	4018		STAA	DSTATE	Now in pos delta state
E7BD	20 00	4019		BRA	82$	
		4020	*			
E7BF	C1 FE	4021	90$	CMPB	#-2	Have we moved down 2 yet?
E7C1	2E D9	4022		BGT	82$	No
E7C3	7D D02B	4023		TST	DSTATE	What state are we in?
E7C6	2C 03	4024		BGE	94$	
		4025	*	BEQ	94$	Undetermined, bump critical count
		4026	*	BPL	94$	We are neg and detected pos delta
E7C8	5F	4027		CLRB		Reset to relative zero point;
E7C9	20 D1	4028		BRA	82$	we are pos and detected pos delta
E7CB	7C D02F	4029	94$	INC	MINMAX	Bump critical count
E7CE	5F	4030		CLRB		Reset to relative zero point
E7CF	86 FF	4031		LDAA	#-1	
E7D1	87 D02B	4032		STAA	DSTATE	Now in neg delta state
E7D4	20 C6	4033		BRA	82$	
		4034	*			
E7D6	B6 D02F	4035	100$	LDAA	MINMAX	Get critical areas detected
E7D9	B1 D033	4036		CMPA	VOCSEN	More than 5 (or 6) reversals detected?
E7DC	2E 02	4037		BGT	GSAM20	Yes, must be voice data
E7DE	4F	4038	GSAM10	CLRA		No, indicate voice not detected
E7DF	39	4039		RTS		
		4040	*			
E7E0	86 01	4041	GSAM20	LDAA	#1	Indicate voice detected
E7E2	39	4042		RTS		
		4043		END		

TABLE 34

```
Error
Address  Code      Sequence                Source Statement
                   2120  *
                   2121  *          Set up for and invoke POISE call progress detection
                   2122  *          This routing requires the dialed ("transfer-to")
                   2123  *          extension number ("X" in the transfer sequence) to be
                   2124  *          passed in the COMRAM.
                   2125  *          Returns:
                   2126  *                A = $01 : ring answered (successful transfer)
                   2127  *                A = $83 : busy or reorder tone
                   2128  *                A = $85 : ring no answer
                   2129  *          The following digits will be passed in COMRAM:
                   2130  *          Terminator (COMRAM byte $00) signifies end of extension
                   2131  *          Zero (COMRAM byte $01) converts to $00 for DTMTON
                   2132  *          One  (COMRAM byte $02) converts to $01 for DTMTON
                   2133  *          etc. etc.
                   2134  *          Nine (COMRAM byte $0A) converts to $09 for DTMTON
                   2135  *          [*]  (COMRAM byte $0B) converts to $0A for DTMTON
                   2136  *          [*]  (COMRAM byte $0C) converts to $0B for DTMTON
                   2137  *          Terminator (COMRAM byte 10) coverts to $0F; end of ext.
                   2138  *
DBAA  96 04        2139  POISED    LDAA   COMSEL  Get the COMRAM which CINDI just set up
DBAC  88 02        2140             EORA   #$02
DBAE  97 04        2141             STAA   COMSEL
DBB0  97 AC        2142             STAA   CA2CB2  Extension number to dial is now in COMRAM
                   2143  *          Note that RSULTC and DODCP will require this bank of COMRAM
DBB2  B6 D032      2144             LDAA   VOCDET  Get iser defined detection threshold
DBB5  B7 D034      2145             STAA   VOCTRK  Set initial voice detect threshold
DBB8  86 03        2146             LDAA   #3      Set initial edge detect to 60-80 ms.
DBBA  B7 D02D      2147             STAA   EDGDET
DBBD  8D 1A        2148             BSR    RSULTC  Do the call progress detection
DBBF  7E D402      2149  2149       JMP    ONLINO  and return status code to CINDI
                   2150  *
                   2151  *          Delay for X milliseconds
                   2152  *          Inputs:  CC = tests the contents of X
                   2153  *                    X  = contains delay in msec
                   2154  *
DBC2  27 14        2155  CPDLAY    BEQ    2$      No need for delay
DBC4  FF D01D      2156             STX    TMR_C1
DBC7  86 01        2157             LDAA   #1      Signal that the timer is going
DBC9  B7 D03E      2158             STAA   CPENBL
DBCC  86 C0        2159             LDAA   #$C0
DBCE  97 AE        2160             STAA   IRQENB
DBD0  FE D01D      2161  1$         LDX    TMR_C1  Wait for delay to end
DBD3  26 FB        2162             BNE    1$
DBD5  BD DDD6      2163             JSR    STOPT1  Stop the timer
DBD8  39           2164  2$         RTS
                   2165  *
```

TABLE 34

```
Error
Address  Code      Sequence  Source Statement

                     2166  *Positive Offhook Indication (POISE) Call Progress Detection
                     2167  *       Inputs:  this routine requires the extension to be in COMRAM
                     2168  *      (so don't switch the COMRAM banks once POISED has set them up)
                     2169  *       Returns:
                     2170  *              A = $01 : ring answered (successful transfer)
                     2171  *                        (directed call pickup received reorder/busy)
                     2172  *              A = $83 : busy or reorder tone
                     2173  *                        (dial tone not received)
                     2174  *              A = $85 : ring no answer
                     2175  *                        (dir. call pickup did not get reorder/busy)
                     2176  *      The status returns seem confusing, but are correct as written.
                     2177  *      Remember that we are performing call progress in response to
                     2178  *      a directed call pickup; based on what it returns we will
                     2179  *      modify the status that we return to the original caller.
                     2180  *      Note that for standard call progress we can use a simple
                     2181  *      timer (INTTMO) to obtain the desired RNA interval;
                     2182  *      for POISE call progress the RNA calculation is rather complex.
                     2183  *      To obtain an RNA interval for POISE that matches INTTMO, we
                     2184  *      must calculate the time between going onhook to drop the line
                     2185  *      and the end of reorder detection.  The reorder detect will
                     2186  *      detect an answer and so should terminate essentially instantly.
                     2187  *      But we will have the following components making up the RNA:
                     2188  *      PBXDTD, 7 second fixed dial tone detect, RNATMO, SHFDUR,
                     2189  *      1 second delay after hookflash, and the time needed to dial
                     2190  *      the Directed Call Pickup sequence (including 200 msec per
                     2191  *      dialed digit, and any W or H embedded in the sequence).  UNTTST
                     2192  *      will calculate a value for RNATMO that will cause all the
                     2193  *      above components to add up to INTTMO.
                     2194  *
DBD9  7D 0036        2195  RSULTC  TST     HUNGUP  Have we already had a hangup interrupt?
DBDC  26 37          2196          BNE     3$      Yes, then fake a successful transfer
                     2197  *       ensure MUX irqs are disabled before going onhook
DBDE  BD D184        2198          JSR     DDDMI   Disable MUX irqs (don't allow a ring irq)
DBE1  BD D17C        2199          JSR     DDDNH   Go onhook
DBE4  FE D05B        2200          LDX     PBXDIS  Delay so that the PBX sees this as a disconnect
DBE7  8D D9          2201          BSR     CPDLAY
DBE9  BD D18C        2202          JSR     DOOFH   Go offhook
                     2203  *       note - may need to check for hangups here (like HOOKIT does)
DBEC  FE D05D        2204          LDX     PBXDTD  Delay before checking for dialtone
DBEF  8D D1          2205          BSR     CPDLAY
                     2206  *       note - if you change the timing of this loop, you must also
                     2207  *       change the calculations of RNATMO in UNTTST.SRC
DBF1  C6 46          2208          LDAB    #70     Test for 7 seconds of dial tone
DBF3  CE 16FA        2209  1$      LDX     #5882   The following loop provides 1/10th second
DBF6  96 A0          2210  2$      LDAA    DTMFCP  Is there noise?
DBF8  84 01          2211          ANDA    #$01
DBFA  27 21          2212          BEQ     5$      No - the noise was interrupted
```

TABLE 34

```
Error
Address  Code      Sequence                Source Statement
DBFC     09        2213          DEX             Count down 1/10th second timer
DBFD     26 F7     2214          BNE     2$
DBFF     5A        2215          DECB            Count down main timer
DC00     26 F1     2216          BNE     1$      (be sure to restart X timer)
                   2217    *     uninterrupted noise detected, indicating dial tone
DC02     FE D05F   2218          LDX     RNATMO  Get the RNA timer and delay for that long
DC05     8D BB     2219          BSR     CPDLAY
DC07     8D 2F     2220          BSR     DDSHF   Do a short hookflash
                   2221    *     note - if you change this timing, you must also change the
                   2222    *     calculations of RNATMO in UNTTST. SRC
DC09     86 0A     2223          LDAA    #10     Delay for 1 second after the flash
DC0B     BD D85C   2224          JSR     DELAY
DC0E     8D 58     2225          BSR     DODCP   Perform a directed call pickup
DC10     BD DCDF   2226          JSR     CK4ROT  Check for re-order/busy tone
DC13     26 04     2227          BNE     4$      ROT absent
                   2228    *     dir. call pickup got reorder/busy; call must have been
                                 transferred ok
DC15     86 01     2229    3$    LDAA    #$01    Flag that successful transfer occurred
DC17     20 1E     2230          BRA     9$      and jam
                   2231    *     dir. call pickup did not get reorder/busy tone; RNA must have
                                 occurred
DC19     86 85     2232    4$    LDAA    #$85    Flag that RNA occurred
DC1B     20 1A     2233          BRA     9$      and jam
                   2234    *     dial tone not detected, indicating that busy or reorder tone
                                 occurred
DC1D     0F        2235    5$    SEI             Disable irqs while we see if caller hungup
DC1E     96 A0     2236          LDAA    DTMFCP  Clear any spurious DTMF/silence interrupts
DC20     86 08     2237          LDAA    #$08    Clear any spurious MUX interrupts
DC22     97 AD     2238          STAA    IRQFLG
DC24     BD D201   2239          JSR     LOPBRK  Is there loop current ? (also reenable HD irqs)
DC27     26 0B     2240          BNE     6$      Loop current present
                   2241    *         Far-end hangup has occurred; deal with it
                   2242    *         (i.e. go onhook, disable silent/DTMF irqs, enable MUX irqs,
                   2243    *         set any flags needed, and put HANGUP key in keys buffer)
DC29     7C 0036   2244          INC     HUNGUP  Count another hangup IRQ
DC2C     BD D4F5   2245          JSR     GRNDIT  Go onhook; select "ring interrupt" as MUX IRQ
DC2F     86 12     2246          LDAA    #HANGUP Put hangup key in buffer
                   2247    *     be sure an SEI command is in effect before doing the JSR INKEY
DC31     BD E70F   2248          JSR     INKEY   Put the key in A into keys buffer
DC34     0E        2249    6$    CLI             Allow interrupts again
DC35     86 83     2250          LDAA    #$83    Flag that Busy or reorder tone occurred
DC37     39        2251    9$    RTS
                   2252    *
                   2253    *           Do a short hookflash
                   2254    *
DC38     F6 D007   2255    DOSHF LDAB    SHFDUR  Get the duration of the short hookflash
DC3B     0F        2256          SEI
```

TABLE 34

Error Address	Code	Sequence		Source Statement		
DC3C	7D 0036	2257		TST	HUNGUP	Have we already had a hangup interrupt?
DC3F	26 25	2258		BNE	1$	Yes - then avoid doing any hookflash
DC41	BD D17C	2259		JSR	DOONH	Put phone onhook
DC44	17	2260		TBA		Place the duration in A (n x 1/10 seconds)
DC45	BD DB5C	2261		JSR	DELAY	
DC48	BD D17C	2262		JSR	DOOFH	Take phone offhook again
DC4B	86 01	2263		LDAA	#1	Give mechanical parts time to work
DC4D	BD DB5C	2264		JSR	DELAY	
DC50	96 A0	2265		LDAA	DTMFCP	Clear any DTMF/silence interrupts
DC52	86 08	2266		LDAA	#$08	Clear any MUX interrupts
DC54	97 AD	2267		STAA	IRQFLG	
DC56	BD D201	2268		JSR	LOPBRK	See if there was a break in loop current
DC59	26 0B	2269		BNE	1$	Loop current present
		2270	*	Far-end hangup has occurred; deal with it		
		2271	*	(i.e. go onhook, disable silent/DTMF irqs, enable MUX irqs,		
		2272	*	set any flags needed, and put hangup key in keys buffer)		
DC5B	7C 0036	2273		INC	HUNGUP	Count another hangup IRQ
DC5E	BD D4F5	2274		JSR	GRNDIT	Go onhook; select "ring interrupt" as MUX IRQ
DC61	86 12	2275		LDAA	#HANGUP	Put hangup key in buffer
		2276	*	be sure an SEI command is in effect before doing the JSR INKEY		
DC63	BD E70F	2277		JSR	INKEY	Put the key in A into keys buffer
DC66	0E	2278	1$	CLI		Allow interrupts again
DC67	39	2279		RTS		
		2280	*			
		2281	*	Dial the directed call pickup sequence		
		2282	*	Inputs: this routing requires the extension to be in COMRAM		
		2283	*	(so don't switch the COMRAM banks once POISED has set them up)		
		2284	*	DCPSEQ is a string of up to 13. bytes, coded as follows:		
		2285	*	H,W,X = IFPSHF, $57, $58 respectively		
		2286	*	digits [0] thru [9] = $80 thru $89		
		2287	*	digits [*],[#] = $8A, $8B respectively		
		2288	*	terminator = FF (so up to 12. bytes are useful)		
		2289	*	The extension will be passed in COMRAM as follows:		
		2290	*	Terminator (COMRAM byte $00) signifies end of extension		
		2291	*	Zero (COMRAM byte $01) converts to $00 for DTMTON		
		2292	*	One (COMRAM byte $02) converts to $01 for DTMTON		
		2293	*	etc. etc.		
		2294	*	Nine (COMRAM byte $0A) converts to $09 for DTMTON		
		2295	*	[*] (COMRAM byte $0B) converts to $0A for DTMTON		
		2296	*	[#] (COMRAM byte $0C) converts to $0B for DTMTON		
		2297	*	Terminator (COMRAM byte $10) converts to $0F ; end of ext.		
		2298	*			
DC68	CE D04E	2299	DODCP	LDX	#DCPSEQ	Index to the directed call pickup sequence
DC6B	A6 00	2300	1$	LDAA	0,X	Get next char in sequence
DC6D	81 FF	2301		CMPA	#$FF	DCP Sequence terminator ?
DC6F	27 41	2302		BEQ	9$	Yes - so jam
DC71	91 28	2303		CMPA	IFPSHF	Is this a hookflash request ?

TABLE 34

Error Address	Code	Sequence				Source Statement
DC73	26 04	2304		BNE	2$	No
DC75	8D C1	2305		BSR	DOSHF	Do a short hookflash
DC77	20 36	2306		BRA	8$	Go on to the next char
DC79	81 57	2307	2$	CMPA	#$57	Is this "W" (1 second wait) ?
DC7B	26 07	2308		BNE	3$	No
DC7D	86 0A	2309		LDAA	#10	Delay 1 second
DC7F	BD DB5C	2310		JSR	DELAY	
DC82	20 2B	2311		BRA	8$	Go to the next char
DC84	81 58	2312	3$	CMPA	#$58	Is this "X" (dial the extension) ?
DC86	26 1C	2313		BNE	7$	No
		2314	**********	dial the extension in COMRAM		
DC88	FF D035	2315		STX	XTEMP	Save our index
DC8B	CE 3000	2316		LDX	#COMRAM	Index the first digit in COMRAM
DC8E	A6 00	2317	5$	LDAA	0,X	Get the next digit
DC90	27 0D	2318		BEQ	6$	COMRAM terminator; we are done dialing extension
DC92	4A	2319		DECA		Remove COMRAM offset
DC93	84 0F	2320		ANDA	#$0F	Strip off upper nibble (just in case)
DC95	81 0F	2321		CMPA	#$0F	Is this a COMRAM terminator?
DC97	27 06	2322		BEQ	6$	Yes
DC99	16	2323		TAB		Pass the char in B to DCPDIL
DC9A	8D 17	2324		BSR	DCPDIL	Dial the digit, and fall into 8$
DC9C	08	2325		INX		Bump to next digit in the extension
DC9D	20 EF	2326		BRA	5$	Keep looping until COMRAM terminator encountered
DC9F	FE D035	2327	6$	LDX	XTEMP	Recover our index
DCA2	20 0B	2328		BRA	8$	and process the next char
		2329	**********			
DCA4	81 8B	2330	7$	CMPA	#$8B	Is this a digit [0]-[9],[*],[#] ?
DCA6	22 07	2331		BHI	8$	No ; ignore out-of-range characters
DCA8	81 80	2332		CMPA	#$80	Is this a digit [0]-[9],[*],[#] ?
DCAA	25 03	2333		BCS	8$	No; ignore out-of-range characters
DCAC	16	2334		TAB		Pass the char in B to DCPDIL
DCAD	8D 04	2335		BSR	DCPDIL	Dial the digit, and fall into 8$
DCAF	08	2336	8$	INX		Bump to next char in sequence
DCB0	20 89	2337		BRA	1$	Keep looping until DCPSEQ terminator encountered
DCB2	39	2338	9$	RTS		
		2339	*			
		2340	*			Dial the digit passed in B
		2341	*			B contains $x0 thru $x9 for digits zero thru nine,
		2342	*			and B contains $xA for [*], and B contains $xB for [#]
		2343	*			Note that DTMTON does not care about the upper nibble.
		2344	*			Be sure to preserve X throughout this routine.
		2345	*			
DCB3	96 AE	2346	DCPDIL	LDAA	IRQENB	Save interrupt enable byte
DCB5	97 02	2347		STAA	XADD2	

TABLE 34

```
Error
Address  Code      Sequence  Source Statement
DCB7     96 3B     2348            LDAA    TONSIL  Save tone/silent enable byte
DCB9     B7 D031   2349            STAA    TMPSIL
DCBC     96 04     2350            LDAA    VIAPC   Save transition settings of MUX and CB1 IRQs
DCBE     97 03     2351            STAA    XADD2+1
DCC0     BD D1D4   2352            JSR     DODTI   Disable tone interrupts
DCC3     BD D1E1   2353            JSR     DODSI   Disable silent interrupts
DCC6     BD D1B4   2354            JSR     DODMI   Disable MUX IRQs
DCC9     17        2355            TBA             Get the tone to issue
DCCA     BD DAAA   2356            JSR     DTMTON  Issue the DTMF tone
DCCD     D6 A0     2357            LDAB    DTMFCP  Clear any tone interrupts generated above
DCCF     D6 03     2358            LDAB    XADD2+1 Restore CB1 and MUX IRQs to former value
DCD1     D7 04     2359            STAB    VIAPC
DCD3     D7 AC     2360            STAB    CA2CB2
DCD5     F6 D031   2361            LDAB    TMPSIL  Restore TONSIL byte to former value
DCD8     D7 3B     2362            STAB    TONSIL
DCDA     D6 02     2363            LDAB    XADD2   Restore IRQ enable byte to former value
DCDC     D7 AE     2364            STAB    IRQENB
DCDE     39        2365            RTS
                   2366    *
                   2367    * Check for reorder/busy tone within a 1 second window
                   2368    * Returns:
                   2369    *       EQ = reorder or busy tone (A = $00)
                   2370    *       NE = not reorder tone, specifically:
                   2371    *            A = $01 : answer/voice detected, hangup occurred
                   2372    *            A = $82 : silence
                   2373    * Note that we do not set up MTRTYP here because we assume this is 'R
                   2374    * type of call progress.  We also do not bother with TMR_C1 (INTTMO and
                   2375    * EXTTMO) since the RNA timer interval is too long for our purposes
                           here.
                   2376    *
DCDF     7D 0036   2377    CK4RDT  TST     HUNGUP  Have we already had a hangup interrupt?
DCE2     27 03     2378            BEQ     1$      No, then we can proceed with call progress
DCE4     86 01     2379            LDAA    #$01    Yes; at any rate, it is not reorder tone
DCE6     39        2380            RTS
DCE7     C6 01     2381    1$      LDAB    #1      Flag that we are doing call progress
DCE9     F7 D03E   2382            STAB    CPENBL
DCEC     C6 C0     2383            LDAB    #$C0    Set bit 6 of IRQENB to enable T1
DCEE     D7 AE     2384            STAB    IRQENB
DCF0     7F D025   2385            CLR     ENERGY  Reset call progress
DCF3     7F D026   2386            CLR     LSTPCK
DCF6     7F D027   2387            CLR     NRGCNT
DCF9     7F 004A   2388            CLR     EDGFLG  Start with no edge detected yet
DCFC     7F D02C   2389            CLR     DTEKT   Clear the voice detection counter
DCFF     86 14     2390            LDAA    #20     Initialize 20 msec timer
DD01     B7 D023   2391            STAA    TMR_C4
DD04     BD D1E1   2392            JSR     DODSI   Disable silent interrupts
DD07     96 38     2393            LDAA    VIARA
```

TABLE 34

Error Address	Code	Sequence		Source Statement		
DD09	8A 02	2394		ORAA	#$02	Force zero/APT high (ready for speech)
DD0B	84 FB	2395		ANDA	#$FB	CVSD direction = input
DD0D	97 38	2396		STAA	VIARA	
DD0F	97 A1	2397		STAA	CVSDIO	
DD11	86 FF	2398		LDAA	#$FF	Set clamp for recording
DD13	B7 1000	2399		STAA	VOLUME	
DD16	86 05	2400		LDAA	#5	Set unknown state voice detect sensitivity
DD18	B7 D033	2401		STAA	VOCSEN	
		2402	*			
DD1B	CE 3200	2403		LDX	#COMRAM+$200	Start of debug data
DD1E	FF D029	2404		STX	DEBUG	
		2405	*			
		2406	*	Wait for energy to begin, then time the energy to determine if ringback,		
		2407	*	busy, or reorder. If >750 msec, assume ringback. Otherwise check for		
		2408	*	noise or errors		
DD21	7F D02E	2409	NRG$	CLR	ERRFLG	Indicate not in error checking
DD24	7F D030	2410		CLR	SAMFLG	Set the no samples flag
DD27	CE 03E8	2411	NR2$	LDX	#1000	Allow 1 second total for edge
DD2A	BD DF2F	2412		JSR	GETEDG	
DD2D	2B 18	2413		BMI	CPM1$	ANSWER (voice) was detected!!
DD2F	26 05	2414		BNE	6$	Some edges occurred
DD31	86 82	2415		LDAA	#$82	No edge in 1 second ... dead line
DD33	7E DD49	2416		JMP	CPMR$	
DD36	7D D025	2417	6$	TST	ENERGY	See if new state is silence
DD39	26 EC	2418		BNE	NR2$	If not go time energy in progress
DD38	86 02	2419		LDAA	#$02	
DD3D	C6 58	2420		LDAB	#88	
DD3F	BD DFDE	2421		JSR	CMXAB	Compare X to 600 msecs
DD42	26 03	2422		BNE	CPM1$	Branch if X>600
DD44	7E DD4E	2423		JMP	CCKER$	<=500 msecs,so check for busy or reorder
		2424	*			
		2425	*		ANSWER DETECTED OR ASSUMED	
		2426	*			
DD47	86 01	2427	CPM1$	LDAA	#1	Assume answer with to message
		2428	*			
		2429	*		End Call Progress Monitoring with status in A reg.	
		2430	*			
DD49	BD DDD6	2431	CPMR$	JSR	STOPT1	Disable T1 and flag "end of call progress"
DD4C	4D	2432		TSTA		Set the EC/NE status appropriately
DD4D	39	2433		RTS		
		2434	*			
		2435	*	Ringback was not detected, but some other cadence or		
		2436	*	noise was. try to identify the cadence . . . it it's noise		
		2437	*	go back and re-monitor. Otherwise, identify the cadence		

TABLE 34

Error Address	Code	Sequence				Source Statement
		2438	*	and return the proper error code. Cadence is determined		
		2439	*	within a 4 second window.		
		2440	*			
DD4E	CE OFA0	2441	CCKER$	LDX	#4000	Check edges within a 4 second window
DD51	FF D01F	2442		STX	TMR_C2	
DD54	7F D028	2443		CLR	EDGCNT	Clear edge count
DD57	CE 023F	2444		LDX	#575	
DD5A	BD DF2F	2445		JSR	GETEDG	
DD5D	2B E8	2446		BMI	CPM1$	ANSWER detected!
DD5F	26 03	2447		BNE	2$	
DD61	7E DD27	2448		JMP	NR2$	Can't be error tone
DD64	7C D028	2449	2$	INC	EDGCNT	Count up an edge
DD67	CE 02EE	2450		LDX	#750	
DD6A	BD DF2F	2451		JSR	GETEDG	
DD6D	2B D8	2452		BMI	CPM1$	ANSWER detected!
DD6F	27 D6	2453		BEQ	CPM1$	Ne edge detected, assume ANSWER
DD71	7F D030	2454		CLR	SAMFLG	Set the no samples flag
DD74	7C D02E	2455		INC	ERRFLG	Indicate into error checking
DD77	FE D01F	2456		LDX	TMR_C2	Timed out
DD7A	26 E8	2457		BNE	2$	No timeout, keep counting
DD7C	7C D028	2458		INC	EDGCNT	One more edge
DD7F	BD DDD6	2459		JSR	STOPT1	Disable T1 and flag "end of call progress"
DD82	B6 D028	2460		LDAA	EDGCNT	How many edges?
DD85	7F S028	2461		CLR	EDGCNT	Clear the edge counter for NRG$
DD88	81 05	2462		CMPA	#5	Check to see if it is busy
DD8A	23 10	2463		BLS	4$	Not likely to be a busy tone if less frequent
DD8C	81 0A	2464		CMPA	#5	Check to see if it is busy
DD8E	22 02	2465		BHI	3$	No . . . may be reorde
DD90	4F	2466		CLRA		Busy signal received, so flag success
DD91	39	2467		RTS		
DD92	91 CD	2468	3$	CMPA	#13	See if in the realm of reorder
DD94	23 06	2469		BLS	4$	Not really, unknown signal
DD96	81 16	2470		CMPA	#22	
DD98	22 02	2471		BHI	4$	Far too frequent for even reorder to be
DD9A	4F	2472		CLRA		Reorder tone was present, so flag success
DD9B	39	2473		RTS		
DD9C	7E DD21	2474	4$	JMP	NRG$	Return on false detection
		2475	*			
		2476	*	Wait for ringback		
		2477	*			
DD9F	86 4E	2478	WT4ANS	LDAA	#'N	Wait for answer, notify after answer detect
DDA1	20 06	2479		BRA	WT4RB1	
DDA3	86 41	2480	WT4ANS	LDAA	#'A	Wait for answer, internal call
DDA5	20 02	2481		BRA	WT4RB1	
DDA7	85 52	2482	WT4RBK	LDAA	#'R	Don't wait for answer, connect on ring
DDA9	36	2483	WT4RB1	PSHA		
DDAA	B6 D032	2484		LDAA	VOCDET	Get user defined detection threshold

TABLE 34

```
Error
Address  Code      Sequence                    Source Statement
DDAD     B7 D034   2485          STAA  VOCHRK  Set initial voice detect threshold
DDB0     86 03     2486          LDAA  #3      Set initial edge detect to 60-80 ms.
DDB2     B7 D02D   2487          STAA  EDGDET
DDB5     FE D017   2488          LDX   PADDLI  Get the pre-answer-detect delay
DDB8     27 13     2489          BEQ   2$      Not need for delay
DDBA     FF D01D   2490          STX   THR_C1
DDBD     86 01     2491          LDAA  #1      Signal that the timer is going
DDBF     B7 D03E   2492          STAA  CPENBL
DDC2     86 C0     2493          LDAA  #$C0
DDC4     97 AE     2494          STAA  CPENBL
DDC6     FE D01D   2495    1$    LDX   THR_C1  Wait for delay to end
DDC9     26 FB     2496          BNE   1$
DDCB     8D 09     2497          BSR   STOPT1  Stop the timer
DDCD     32        2498    2$    PULA
DDCE     FE D009   2499          LDX   INTTMO  Timeout for call prog-internal-pass in X
DDD1     8D 12     2500          BSR   RESULT  Perform call progress detection
DDD3     7E D402   2501          JMP   ONLINO  and return call progress status code to CINDI
                   2502    *
                   2503    *     Disable the T1 timer because call progress finished.
                   2504    *     Be careful never to trash A register in here.
                   2505    *     Trashes:  B register
                   2506    *
DDD6     C6 40     2507    STOPT1 LDAB  #$40     Disable T1 timer
DDD8     D7 AE     2508          STAB  IRHENB
DDDA     5F        2509          CLRB
DDDB     F7 D03E   2510          STAB  CPENBL    Flag that we are not longer doing call progress
DDDE     F7 D01B   2511          STAB  THR_OH
DDE1     F7 D01C   2512          STAB  THR_OH+1
                                                Clear the on-hook timer also
DDE4     39        2513          RTS
                   2514    *
```

What is claimed is:

1. A voice processing system, including:

a digital bus;

a communications interface unit coupled to the bus and adapted to be connected to a telephone line, the communications interface unit including a means for digitizing voice signals received on the telephone line and transferring the digitized voice signals to the bus; and

a general purpose central processing unit coupled to the bus, and including a means for receiving the digitized voice signals from the bus, and a means for processing digital signals, including the digitized voice signals received from the bus.

2. The system of claim 1, wherein the central processing unit is capable of processing both digitized voice signals and other digital signals in a single record.

3. The system of claim 1, wherein the central processing unit is capable of controlling the operation of the communications interface unit by transmitting digital instructions to the communications interface unit over the digital bus.

4. The system of claim 1, wherein the communications interface unit includes means for converting a digitized voice signal into an analog voice signal, and wherein the central processing unit is capable of transmitting a first digitized voice signal and a set of digital instructions to the communications interface unit over the digital bus, to cause the communications interface unit to convert the first digitized voice signal to a first analog voice signal and transmit the first analog voice signal over the telephone line.

5. The system of claim 1, wherein the digital bus is a VME bus.

6. The system of claim 1, wherein the central processing unit includes an initiator for establishing communication between the central processing unit and a target device coupled to the central processing unit by a Small Computer System Interface.

7.The system of claim 6, also including means for facilitating a virtual port access of the target device.

8. A voice processing system, including:

a digital bus;

a communications interface unit coupled to the bus and adapted to be connected to a telephone line, the communications interface unit including a means for digitizing voice signals received on the telephone line and transferring the digitized voice signals to the bus; and

a general purpose central processing unit coupled to the bus, and including a memory unit and a means for receiving the digitized voice signals from the bus, wherein the central processing unit is programmed to create files for storing the digitized voice signals in the memory unit.

9. The system of claim 8, wherein the central processing unit is programmed to include a cache memory, and to create a pseudo-file in the cache memory for each digitized voice signal received from the bus, and is programmed to write each said digitized voice signal received from the bus into the memory unit using

information in said pseudo-file, and then to write the information in the pseudo-file into the memory unit.

10. The system of claim 8, wherein the central processing unit is capable of generating a digitized voice menu signal, wherein the communications interface unit includes means for converting the digitized voice menu signal into an analog voice menu signal, and wherein the central processing unit is capable of transmitting the digitized voice menu signal and a set of digital instructions to the communications interface unit over the digital bus, to cause the communications interface unit to convert the digitized voice menu signal into the analog voice menu signal and transmit the analog voice menu signal over the telephone line.

11. The system of claim 10, wherein the central processing unit is capable of modifying the digitized voice menu signal and transmitting the modified digitized voice menu signal and a second set of digital instructions to the communications interface unit over the digital bus, to cause the communications interface unit to convert the modified digitized voice menu signal into a second analog voice menu signal and to transmit the second analog voice menu signal over the telephone line.

12. The system of claim 10, wherein the digital voice menu signal includes digitized voice phrases.

13. The system of claim 10, wherein the analog voice menu signal prompts a listener on the telephone line to instruct the central processing unit to execute an application program, wherein the communications interface unit is capable of receiving a digital command

signal on the telephone line from the listener and forwarding the digital command signal to the central processing unit over the bus, and wherein the central processing unit is programmed to execute the application program in response to the digital command signal.

14. The system of claim 13, wherein the central processing unit is programmed to cause the communications interface unit to transmit interrogating voice signals to the listener during execution of the application program, which interrogating voice signals prompt the listener to transmit an answer signal over the telephone line.

15. The system of claim 14, wherein the communications interface unit is capable of capable of receiving the answer signal from the telephone line and forwarding the answer signal to the central processing unit over the bus, and wherein the central processing unit is programmed to generate a digital report signal in response to the answer signal.

16. The system of claim 8, wherein the central processing unit is capable of generating a tagged message signal and transmitting the tagged message signal to the communications interface unit over the bus.

17. The system of claim 16, wherein the tagged message signal includes a data signal and a prompt signal for instructing the communications interface unit to transmit a prompting message over the telephone line.

18. The system of claim 17, wherein the data signal includes a digitized voice message, wherein the

communications interface unit is capable of converting the digitized voice message into an analog voice signal and transmitting said analog voice signal over the telephone line in response to receipt of the tagged message, and wherein the communications interface unit is capable of transmitting said prompting message over the telephone line in response to receipt of the tagged message.

19. The system of claim 18, wherein the tagged message includes an execute prompt signal for instructing the communications interface unit to transmit an execute prompt message over the telephone line, and wherein the communications interface unit is capable of converting the execute prompt signal into an analog execute prompt message and transmitting said analog execute prompt message over the telephone line in response to receipt of the tagged message.

20. The system of claim 19, wherein said analog execute prompt message prompts a listener on the telephone line to instruct the central processing unit to execute an application program.

21. The system of claim 8, wherein the communications interface unit is programmed to initiate a telephone call in response to a telephone station request signal transmitted by a caller over the telephone line, and to monitor the progress of the telephone call.

22. The system of claim 8, wherein the communications interface unit is programmed to execute a blind telephone call transfer in response to a transfer request signal transmitted from a caller station by a caller over the telephone line, and to go off hook and

then detect the status of the remote end of the telephone line promptly after executing the blind transfer.

23. The system of claim 22, wherein the communications interface unit is programmed to transmit a first voice message over the telephone line upon determining that there is no dial tone being generated by the remote end of the telephone line.

24. The system of claim 22, wherein the communications interface unit is programmed to wait for a predetermined period of time upon determining that a dial tone is being generated by the remote end of the telephone line, and then executing a directed call pickup operation at the end of said predetermined period of time.

25. The system of claim 24, wherein the communications interface unit is programmed to transmit a second voice message over the telephone line if it detects no reorder or error tone following execution of said directed call pickup operation.

26. The system of claim 1, wherein the communications interface unit is adapted to be connected to a pair of telephone lines.

27. The system of claim 1, also including a second communications interface unit coupled to the bus and adapted to be connected to a second telephone line, the second communications interface unit including a means for digitizing voice signals received on the second telephone line and transferring the digitized voice signals to the bus.

28. A voice processing method, including the steps of:

digitizing voice signals received on a telephone line and transmitting the digitized voice signals to a digital bus;

receiving the digitized voice signals from the bus, and supplying them to a central processing unit; and processing both the digitized voice signals and other digital signals in a single record in the central processing unit.

29. The method of claim 28, also including the steps of:

transmitting a first digitized voice signal and a set of digital instructions to a communications interface unit over the bus;

converting the first digitized voice signal to a first analog voice signal; and

transmitting the first analog voice signal over the telephone line.

30. The method of claim 28, also including the steps of:

storing the digitized voice signals in files in a memory unit.

31. The method of claim 30, also including the steps of:

creating a pseudo-file in a cache memory for each digitized voice signal received from the bus;

writing each said digitized voice signal received from the bus into a memory unit using information in said pseudo-file; and

thereafter, writing the information in the pseudo-file into the memory unit.

32. The method of claim 29, also including the steps of:

generating a digitized voice menu signal;

converting the digitized voice menu signal into an analog voice menu signal;

transmitting the digitized voice menu signal and a set of digital instructions to the communications interface unit over the digital bus; and

converting the digitized voice menu signal into the analog voice menu signal and transmitting the analog voice menu signal over the telephone line.

33. The method of claim 32, also including the steps of:

modifying the digitized voice menu signal and transmitting the modified digitized voice menu signal and a second set of digital instructions to the communications interface unit over the digital bus; and

converting the modified digitized voice menu signal into a second analog voice menu signal and transmitting the second analog voice menu signal over the telephone line.

34. The method of claim 32, wherein the analog voice menu signal prompts a listener on the telephone line to command the central processing unit to execute an application program, and including the steps of:

receiving a digital command signal on the telephone line from the listener and forwarding the digital command signal to the central processing unit over the bus; and

executing the application program in response to the digital command signal.

35. The method of claim 34, also including the steps of:

transmitting interrogating voice signals to the listener during execution of the application program,

which interrogating voice signals prompt the listener to transmit an answer signal over the telephone line.

36. The method of claim 35, also including the steps of:

receiving the answer signal from the telephone line and forwarding the answer signal to the central processing unit over the bus; and

generating a digital report signal in response to the answer signal.

37. The method of claim 29, also including the steps of:

generating a tagged message signal and transmitting the tagged message signal to the communications interface unit over the bus.

38. The method of claim 36, wherein the tagged message signal includes a data signal and a prompt signal for instructing the communications interface unit to transmit a prompting message over the telephone line.

39. The method of claim 38, wherein the data signal includes a digitized voice message, and also including the steps of:

converting the digitized voice message into an analog voice signal and transmitting said analog voice signal over the telephone line in response to receipt of the tagged message; and

transmitting said prompting message over the telephone line in response to receipt of the tagged message.

40. The method of claim 39, wherein the tagged message includes an execute prompt signal for instructing the communications interface unit to transmit an execute

prompt message over the telephone line, and also including the steps of:

converting the execute prompt signal into an analog execute prompt message and transmitting said analog execute prompt message over the telephone line in response to receipt of the tagged message.

41. The method of claim 40, wherein said analog execute prompt message prompts a listener on the telephone line to instruct the central processing unit to execute an application program.

42. The method of claim 29, also including the steps of:

initiating a telephone call in response to a telephone station request signal transmitted by a caller over the telephone line; and

monitoring the progress of the telephone call.

43. The method of claim 29, also including the steps of:

executing a blind telephone call transfer in response to a transfer request signal transmitted from a caller station by a caller over the telephone line; and

then detecting the status of the remote end of the telephone line promptly after executing the blind transfer.

44. The method of claim 43, also including the steps of:

transmitting a first voice message over the telephone line upon determining that there is no dial tone being generated by the remote end of the telephone line.

45. The method of claim 44, also including the steps of:

waiting for a predetermined period of time upon determining that a dial tone is being generated by the remote end of the telephone line, and then executing a directed call pickup operation at the end of said predetermined period of time.

46. The method of claim 45, also including the step of:

transmitting a second voice message over the telephone line upon failing to detect a reorder or error tone after executing said directed call pickup operation.

FIG .1A

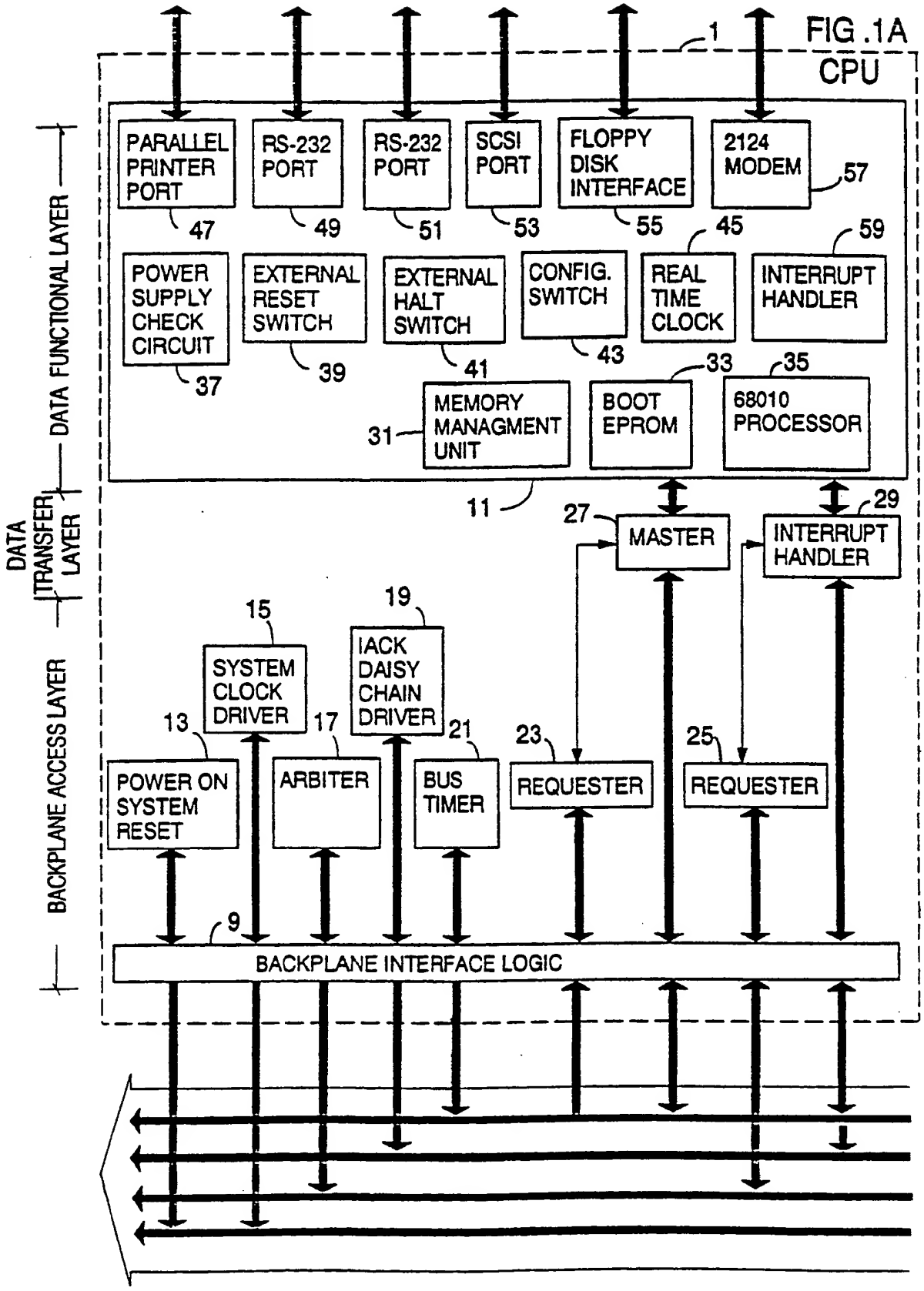
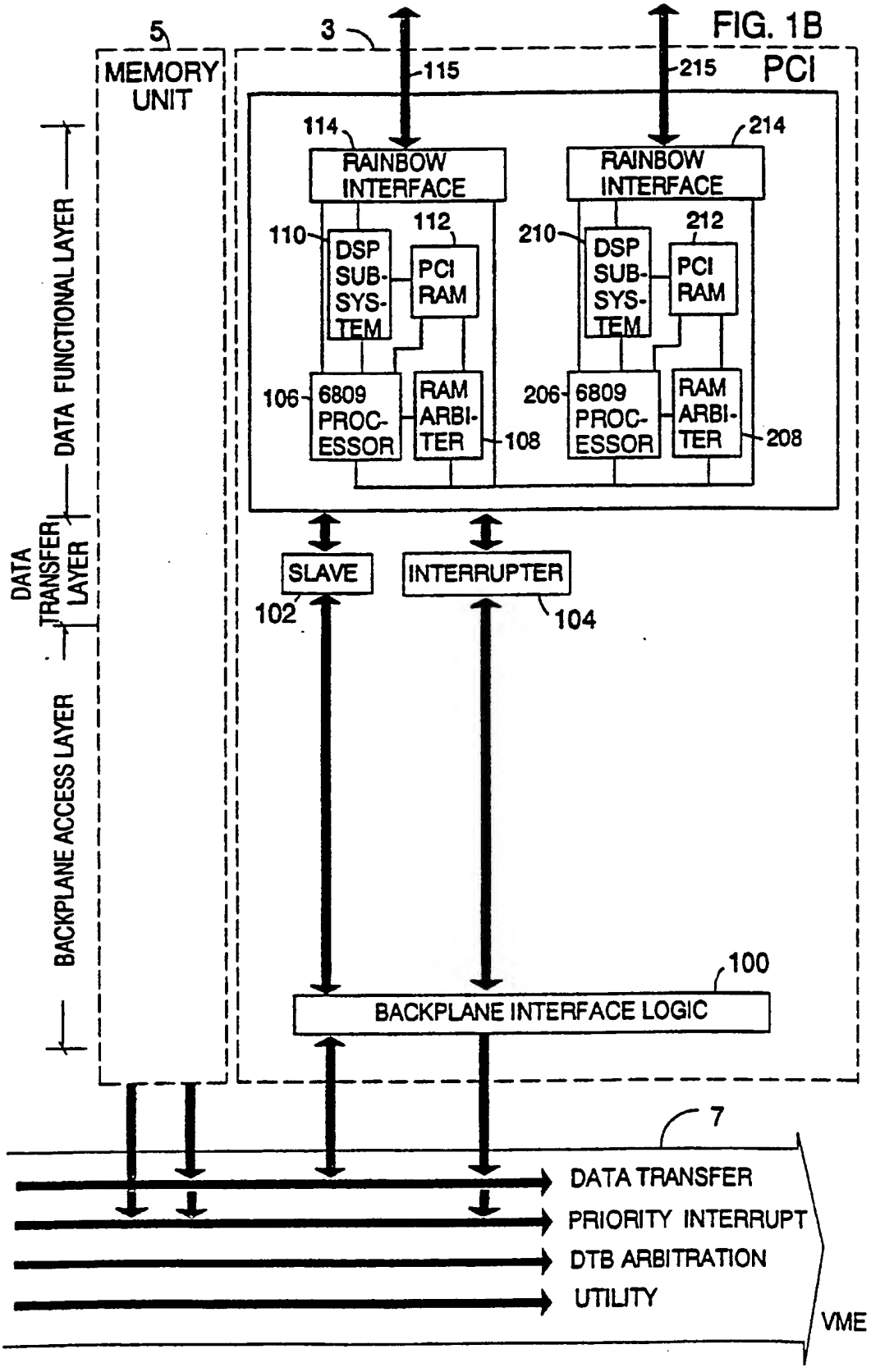

1
CPU
PARALLEL PRINTER PORT
47
RS-232 PORT
49
RS-232 PORT
51
SCSI PORT
53
FLOPPY DISK INTERFACE
55
2124 MODEM
57
45
59
POWER SUPPLY CHECK CIRCUIT
37
EXTERNAL RESET SWITCH
39
EXTERNAL HALT SWITCH
41
CONFIG. SWITCH
43
REAL TIME CLOCK
INTERRUPT HANDLER
33
35
MEMORY MANAGMENT UNIT
31
BOOT EPROM
68010 PROCESSOR
11
27
MASTER
INTERRUPT HANDLER
29
DATA TRANSFER LAYER
DATA FUNCTIONAL LAYER
BACKPLANE ACCESS LAYER
15
SYSTEM CLOCK DRIVER
19
IACK DAISY CHAIN DRIVER
13
17
21
23
25
POWER ON SYSTEM RESET
ARBITER
BUS TIMER
REQUESTER
REQUESTER
9
BACKPLANE INTERFACE LOGIC

FIG. 1B



SUBSTITUTE SHEET

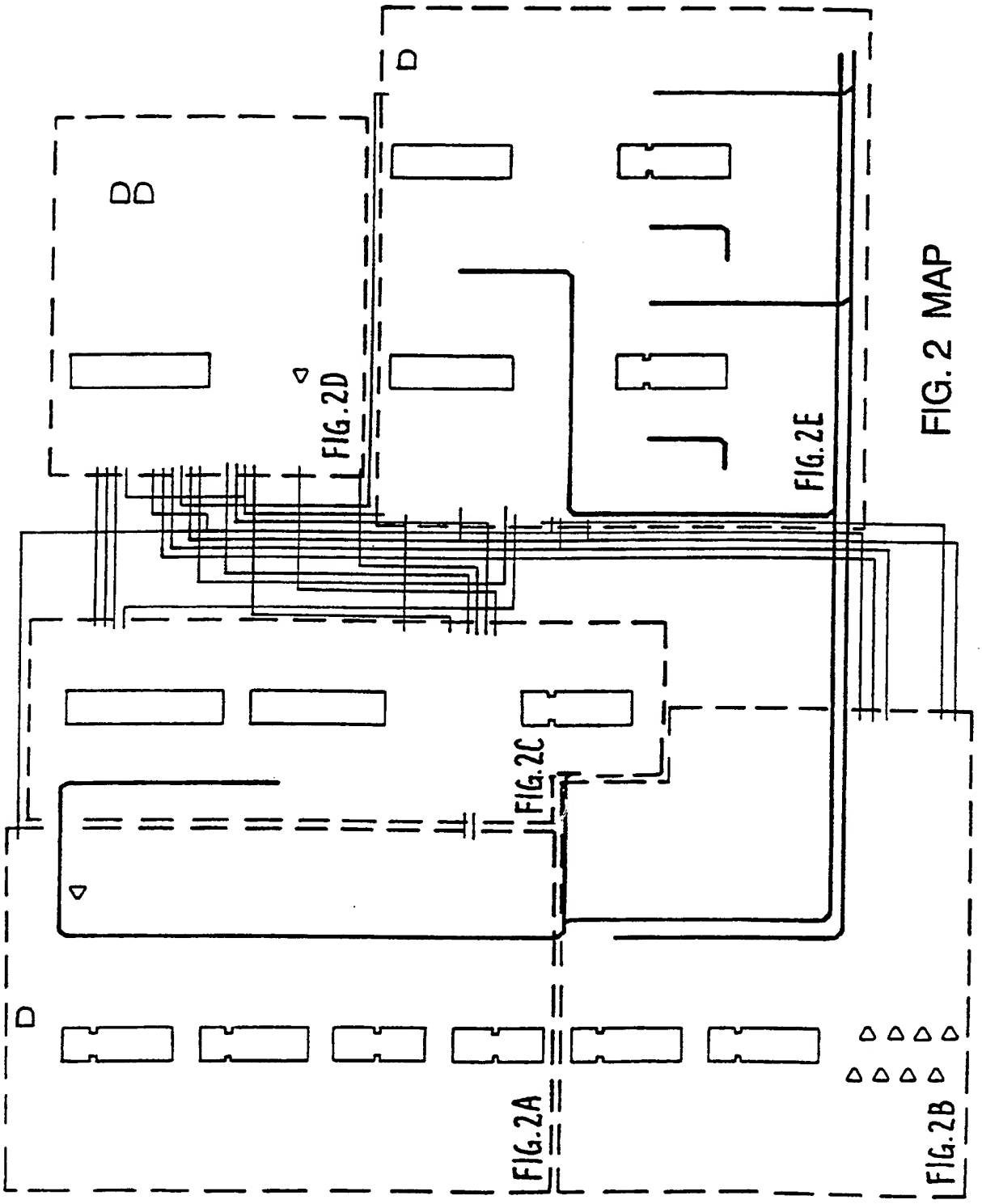


FIG. 2 MAP

SHEET

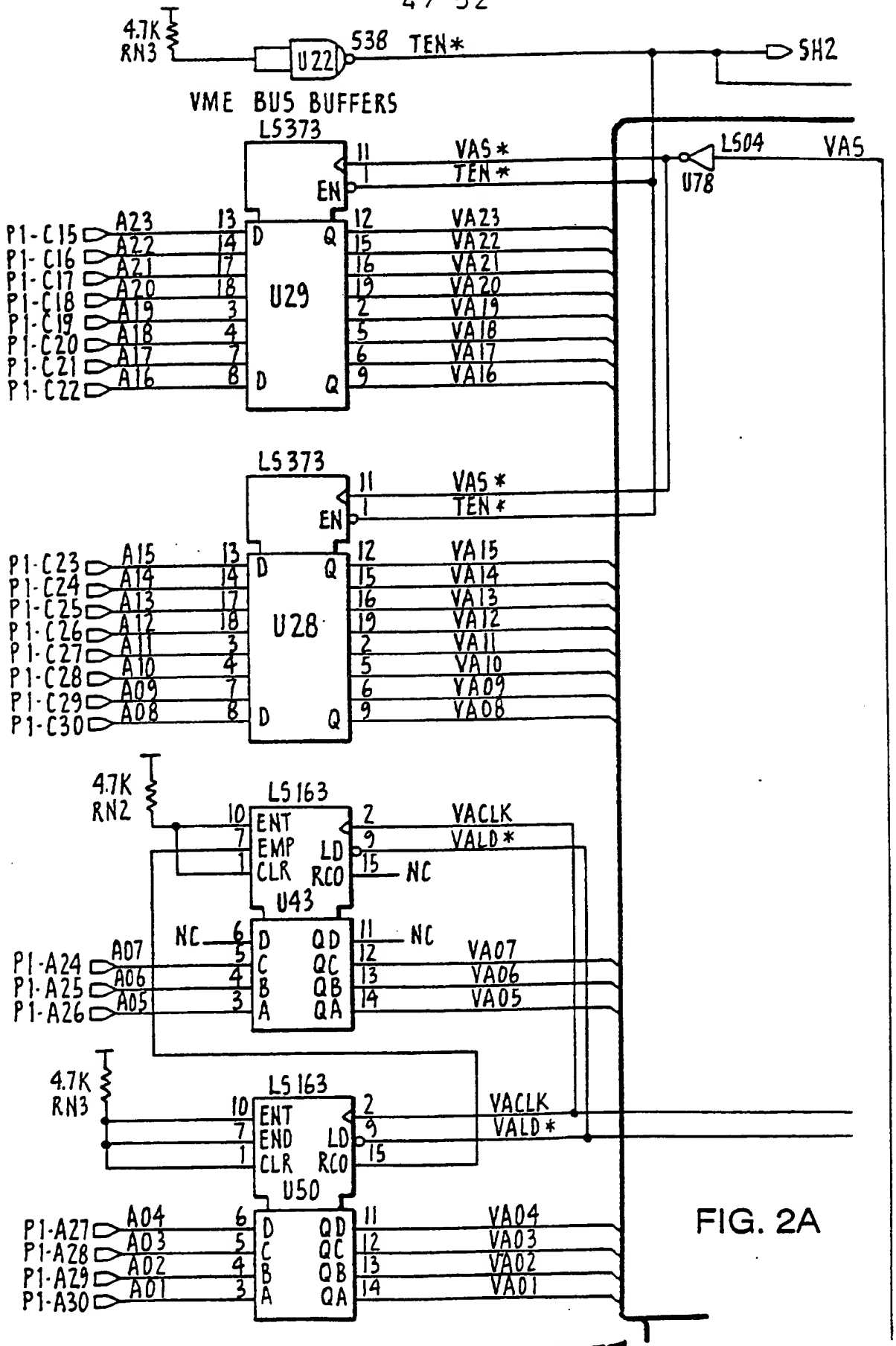

4.7K
RN3
U22
538
TEN*
SH2
VME BUS BUFFERS
LS373
EN
VAS*
TEN*
LS04
U78
VAS
U29
A23
A22
A21
A20
A19
A18
A17
A16
P1-C15
P1-C16
P1-C17
P1-C18
P1-C19
P1-C20
P1-C21
P1-C22
VA23
VA22
VA21
VA20
VA19
VA18
VA17
VA16
LS373
EN
VAS*
TEN*
U28
A15
A14
A13
A12
A11
A10
A09
A08
P1-C23
P1-C24
P1-C25
P1-C26
P1-C27
P1-C28
P1-C29
P1-C30
VA15
VA14
VA13
VA12
VA11
VA10
VA09
VA08
4.7K
RN2
LS163
ENT
EMP
CLR
LD
RCO
U43
VACLK
VALD*
NC
NC
A07
A06
A05
P1-A24
P1-A25
P1-A26
QD
QC
QB
QA
VA07
VA06
VA05
4.7K
RN3
LS163
ENT
END
CLR
LD
RCO
U50
VACLK
VALD*
A04
A03
A02
A01
P1-A27
P1-A28
P1-A29
P1-A30
VA04
VA03
VA02
VA01

FIG. 2A

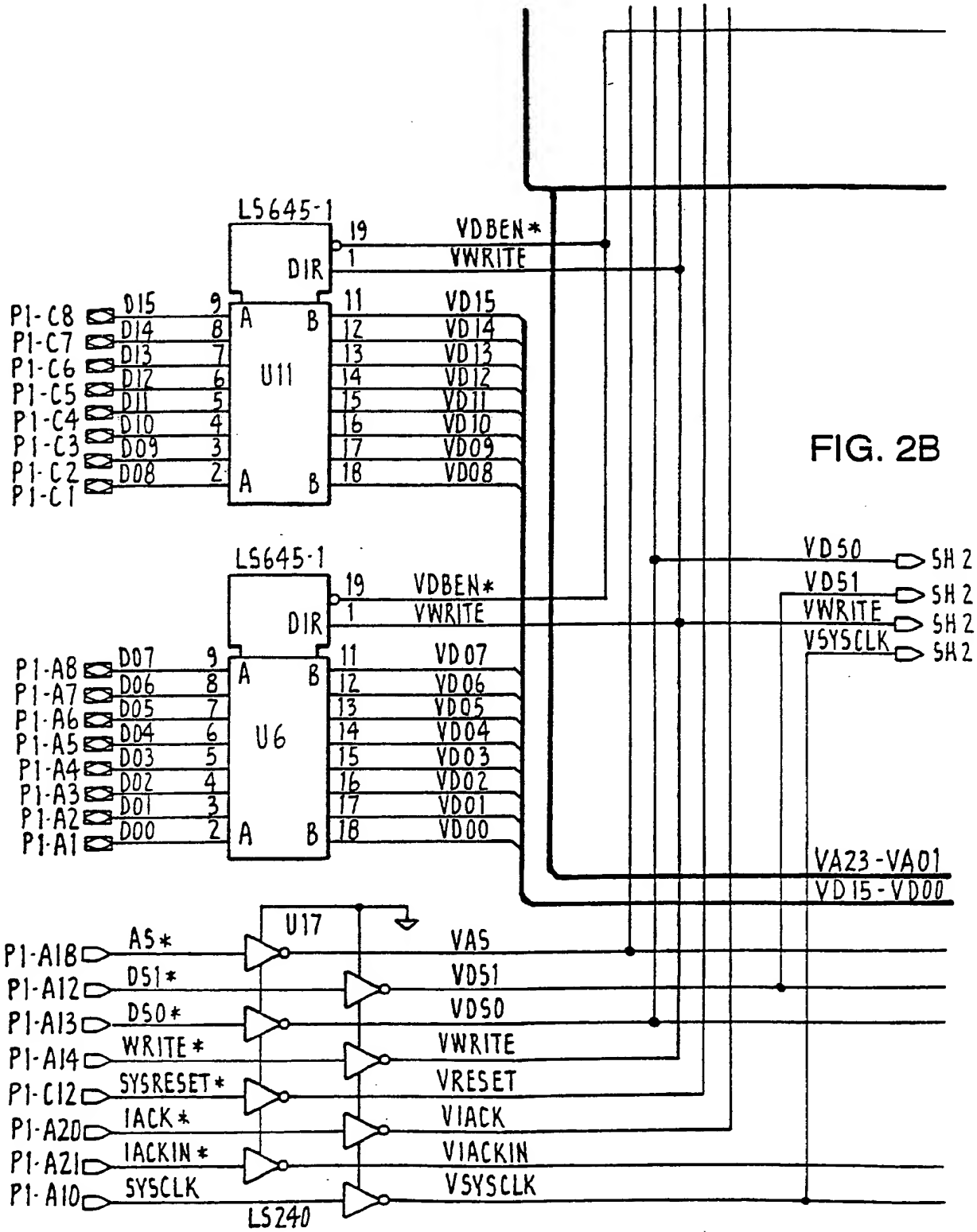

LS645-1
DIR
19
1
VDBEN*
VWRITE
U11
A
B
P1-C8
P1-C7
P1-C6
P1-C5
P1-C4
P1-C3
P1-C2
P1-C1
D15
D14
D13
D12
D11
D10
D09
D08
9
8
7
6
5
4
3
2
11
12
13
14
15
16
17
18
VD15
VD14
VD13
VD12
VD11
VD10
VD09
VD08
LS645-1
DIR
19
1
VDBEN*
VWRITE
U6
P1-A8
P1-A7
P1-A6
P1-A5
P1-A4
P1-A3
P1-A2
P1-A1
D07
D06
D05
D04
D03
D02
D01
D00
VD07
VD06
VD05
VD04
VD03
VD02
VD01
VD00
VD50
VD51
VWRITE
VSYSCLK
SH 2
VA23-VA01
VD15-VD00
U17
P1-A18
P1-A12
P1-A13
P1-A14
P1-C12
P1-A20
P1-A21
P1-A10
AS*
DS1*
DS0*
WRITE*
SYSRESET*
IACK*
IACKIN*
SYSCLK
VAS
VDS1
VDS0
VWRITE
VRESET
VIACK
VIACKIN
VSYSCLK
LS240

FIG. 2B

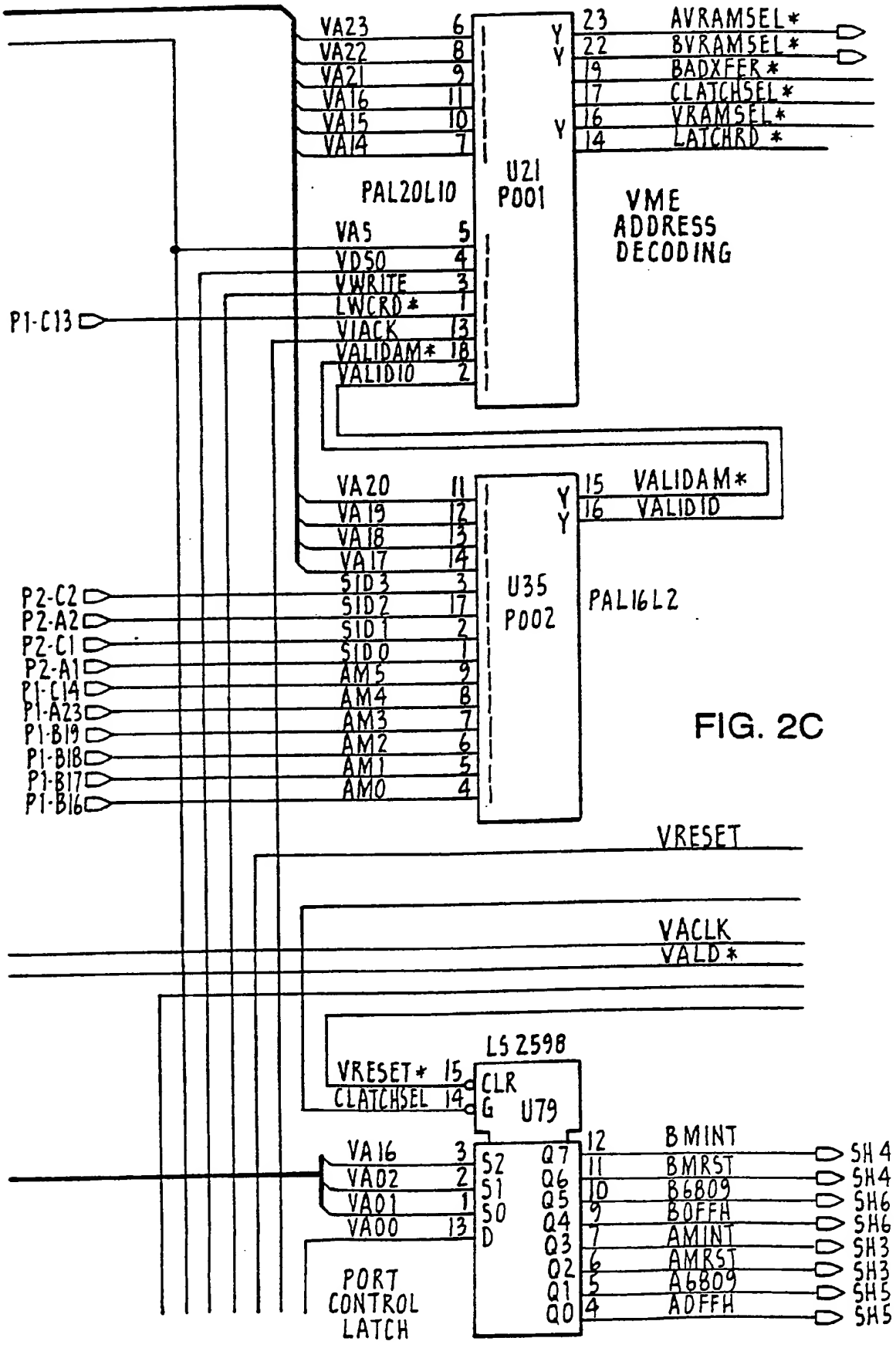

VA23 6
VA22 8
VA21 9
VA16 11
VA15 10
VA14 7
PAL20L10
U21
P001
VME
ADDRESS
DECODING
VA5 5
VD50 4
VWRITE 3
LWCRD* 1
VIACK 13
VALIDAM* 18
VALIDIO 2
Y 23 AVRAMSEL*
Y 22 BVRAMSEL*
19 BADXFER*
17 CLATCHSEL*
Y 16 VRAMSEL*
14 LATCHRD*
P1-C13
VA20 11
VA19 12
VA18 13
VA17 14
SID3 3
SID2 17
SID1 2
SID0 1
AM5 9
AM4 8
AM3 7
AM2 6
AM1 5
AM0 4
U35
P002
PAL16L2
Y 15 VALIDAM*
Y 16 VALIDIO
P2-C2
P2-A2
P2-C1
P2-A1
P1-C14
P1-A23
P1-B19
P1-B18
P1-B17
P1-B16
VRESET
VACLK
VALD*
LS 2598
VRESET* 15
CLATCHSEL 14
CLR
G
U79
VA16 3
VA02 2
VA01 1
VA00 13
S2
S1
S0
D
Q7 12 BMINT
Q6 11 BMRST
Q5 10 B6809
Q4 9 BOFFH
Q3 7 AMINT
Q2 6 AMRST
Q1 5 A6809
Q0 4 ADFFH
SH4
SH4
SH6
SH6
SH3
SH3
SH5
SH5
PORT
CONTROL
LATCH

FIG. 2C

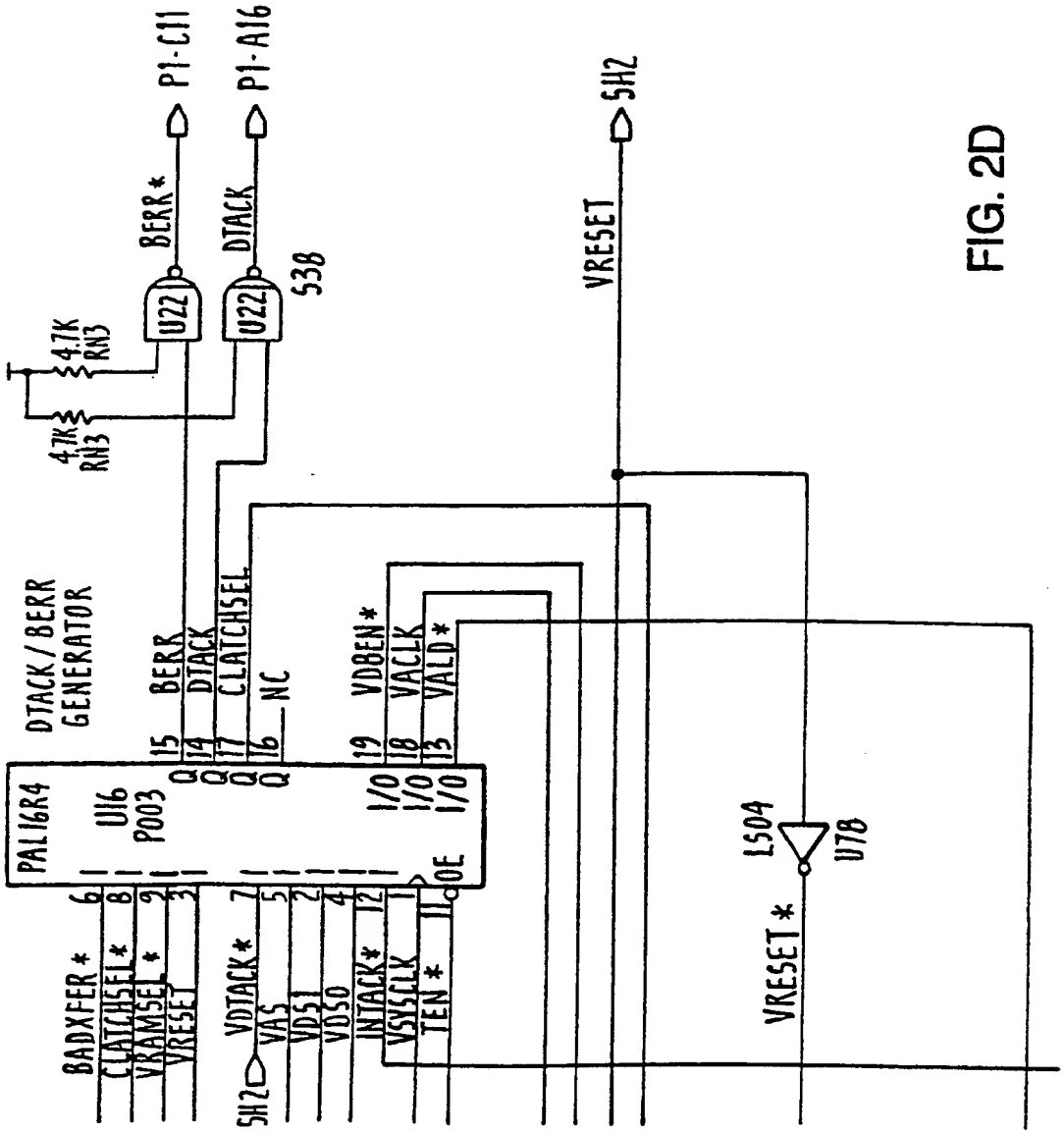

P1-C11
P1-A16
SH2
BERR*
DTACK
VRESET
U22
U22
538
4.7K RN3
4.7K RN3
DTACK/BERR GENERATOR
PAL16R4
U16 P003
BERR
DTACK
CLATCHSEL
NC
VDBEN*
VACLK
VALD*
I/O
I/O
I/O
OE
BADXFER*
CLATCHSEL*
VRAMSEL*
VRESET
VDTACK*
VAS
VD31
VD30
INTACK*
VSYSCLK
TEN*
SH2
LS04
U78
VRESET*

FIG. 2D

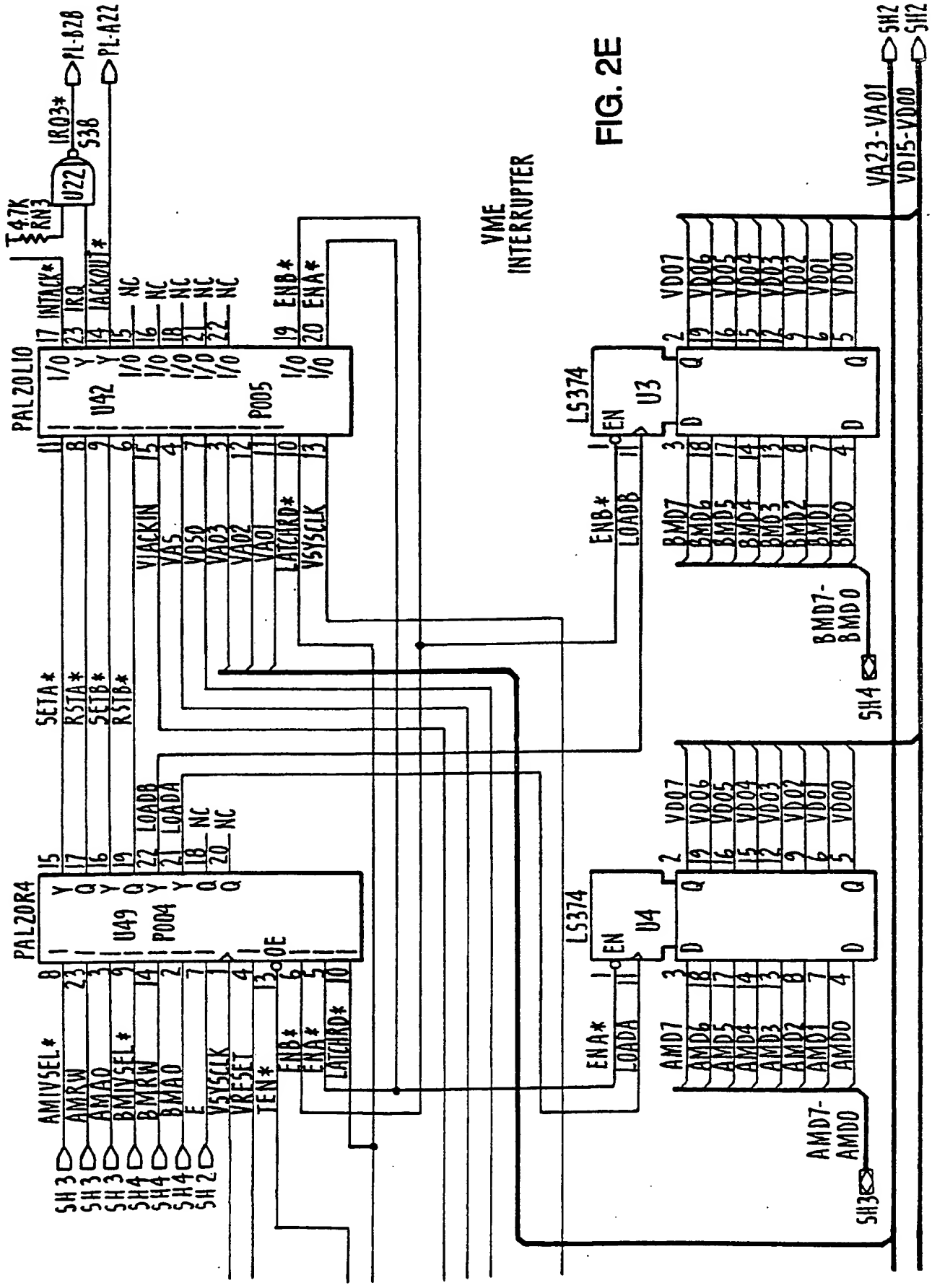


FIG. 2E

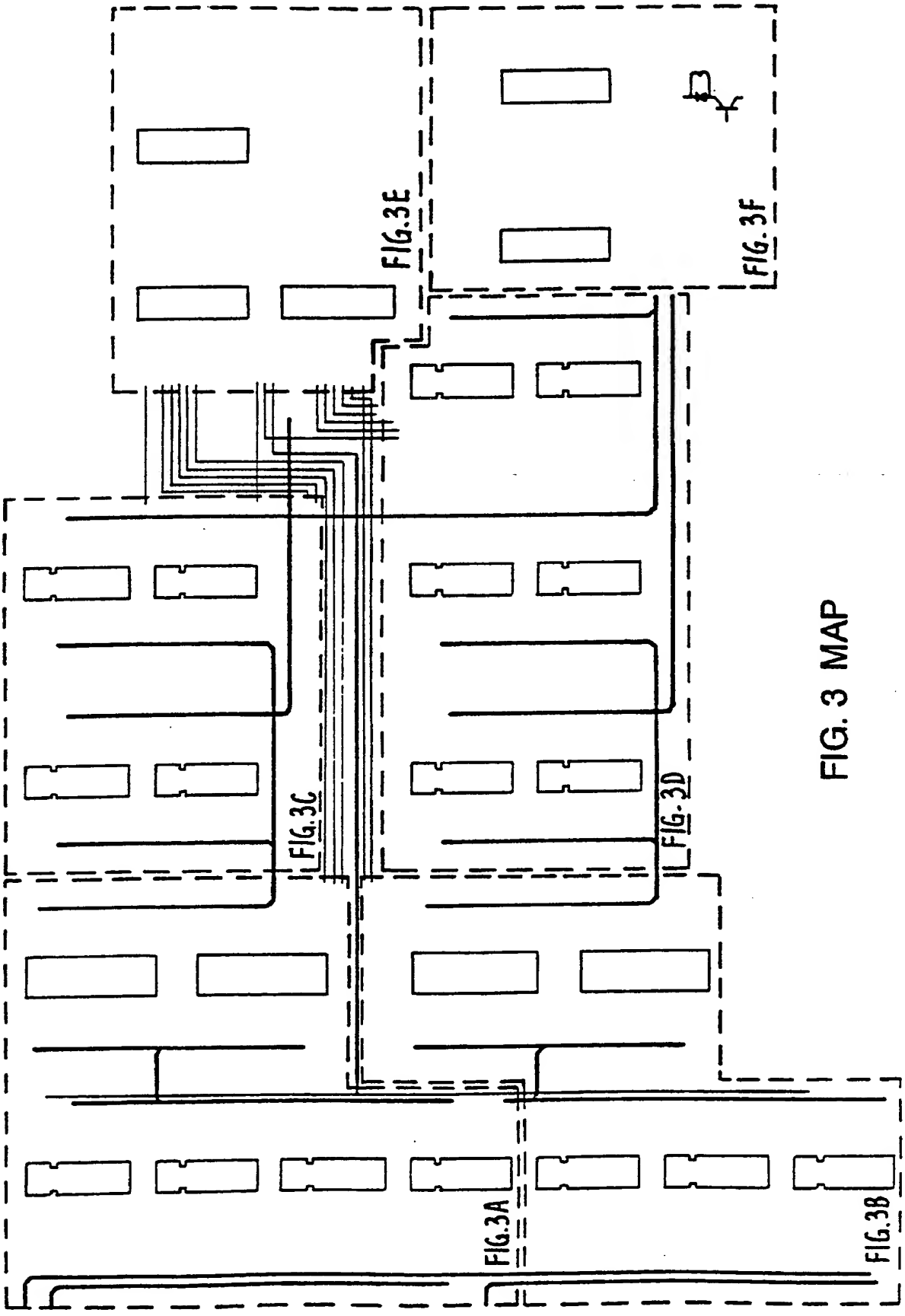


FIG. 3 MAP

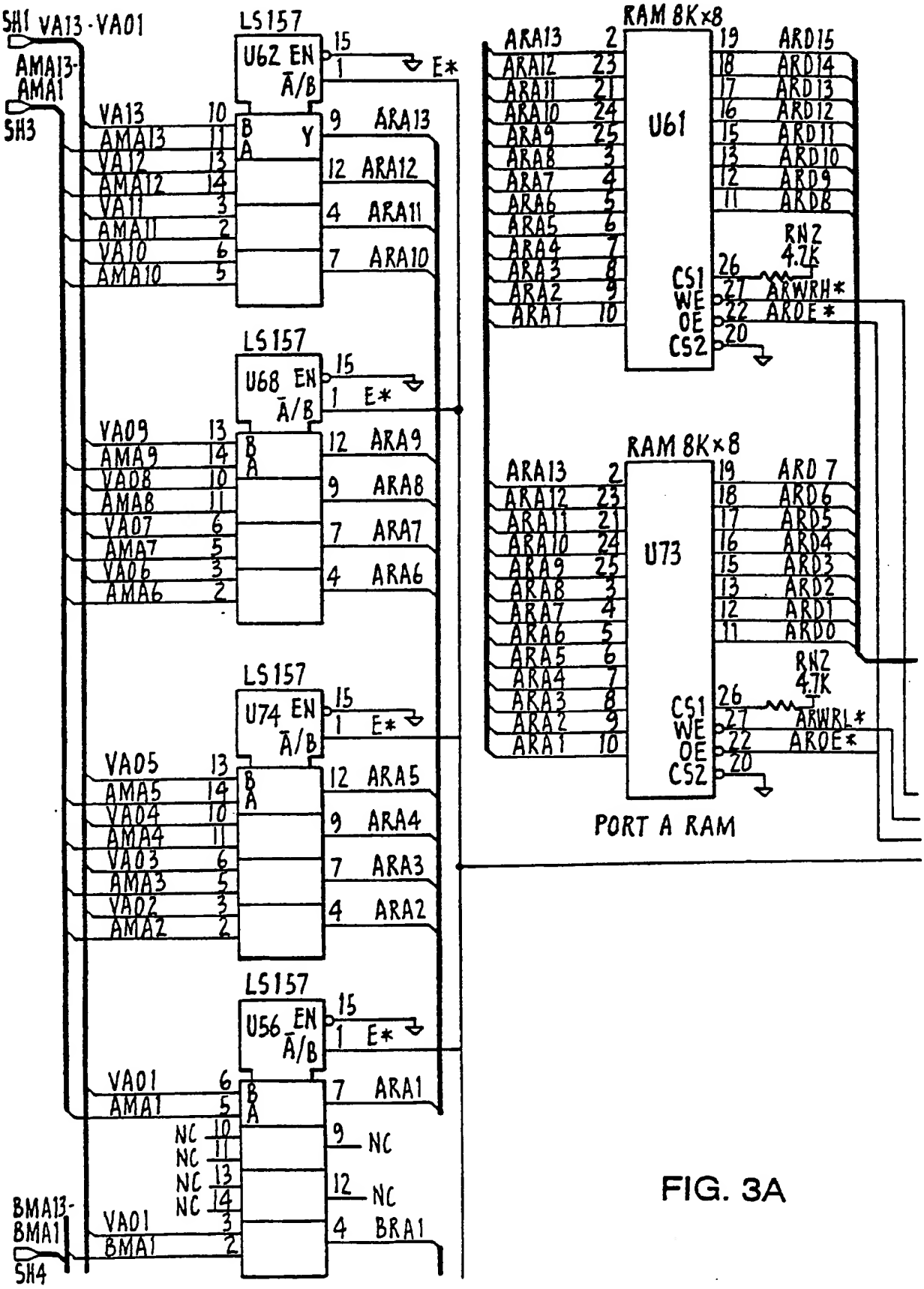

SH1
VA13-VA01
AMA13-AMA1
SH3
LS157
U62 EN Ā/B
E*
VA13
AMA13
VA12
AMA12
VA11
AMA11
VA10
AMA10
ARA13
ARA12
ARA11
ARA10
LS157
U68 EN Ā/B
E*
VA09
AMA9
VA08
AMA8
VA07
AMA7
VA06
AMA6
ARA9
ARA8
ARA7
ARA6
LS157
U74 EN Ā/B
E*
VA05
AMA5
VA04
AMA4
VA03
AMA3
VA02
AMA2
ARA5
ARA4
ARA3
ARA2
LS157
U56 EN Ā/B
E*
VA01
AMA1
ARA1
NC
NC
NC
NC
BMA13-BMA1
SH4
VA01
BMA1
BRA1
RAM 8K×8
U61
ARA13
ARA12
ARA11
ARA10
ARA9
ARA8
ARA7
ARA6
ARA5
ARA4
ARA3
ARA2
ARA1
CS1
WE
OE
CS2
ARD15
ARD14
ARD13
ARD12
ARD11
ARD10
ARD9
ARD8
RN2 4.7K
ARWRH*
AROE*
RAM 8K×8
U73
ARD 7
ARD6
ARD5
ARD4
ARD3
ARD2
ARD1
ARD0
RN2 4.7K
ARWRL*
AROE*
PORT A RAM

FIG. 3A

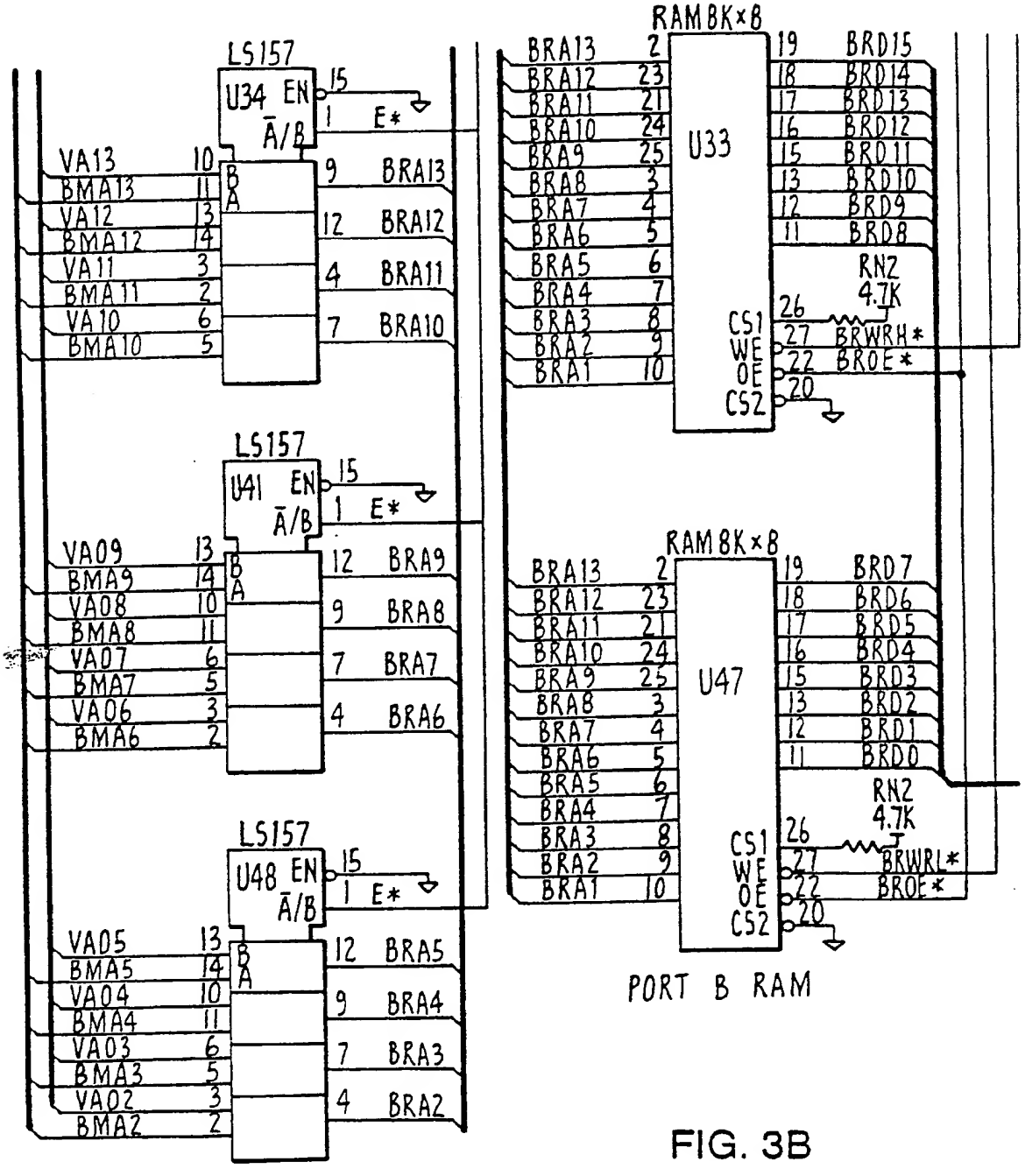

LS157
U34 EN
Ā/B
E*
VA13
BMA13
VA12
BMA12
VA11
BMA11
VA10
BMA10
BRA13
BRA12
BRA11
BRA10
LS157
U41 EN
Ā/B
E*
VA09
BMA9
VA08
BMA8
VA07
BMA7
VA06
BMA6
BRA9
BRA8
BRA7
BRA6
LS157
U48 EN
Ā/B
E*
VA05
BMA5
VA04
BMA4
VA03
BMA3
VA02
BMA2
BRA5
BRA4
BRA3
BRA2
RAM 8K×8
U33
BRA13
BRA12
BRA11
BRA10
BRA9
BRA8
BRA7
BRA6
BRA5
BRA4
BRA3
BRA2
BRA1
BRD15
BRD14
BRD13
BRD12
BRD11
BRD10
BRD9
BRD8
RNZ
4.7K
CS1
WE
OE
CS2
BRWRH*
BROE*
RAM 8K×8
U47
BRD7
BRD6
BRD5
BRD4
BRD3
BRD2
BRD1
BRD0
RN2
4.7K
BRWRL*
BROE*
PORT B RAM
RAM ADDRESS MULTIPLEXERS

FIG. 3B

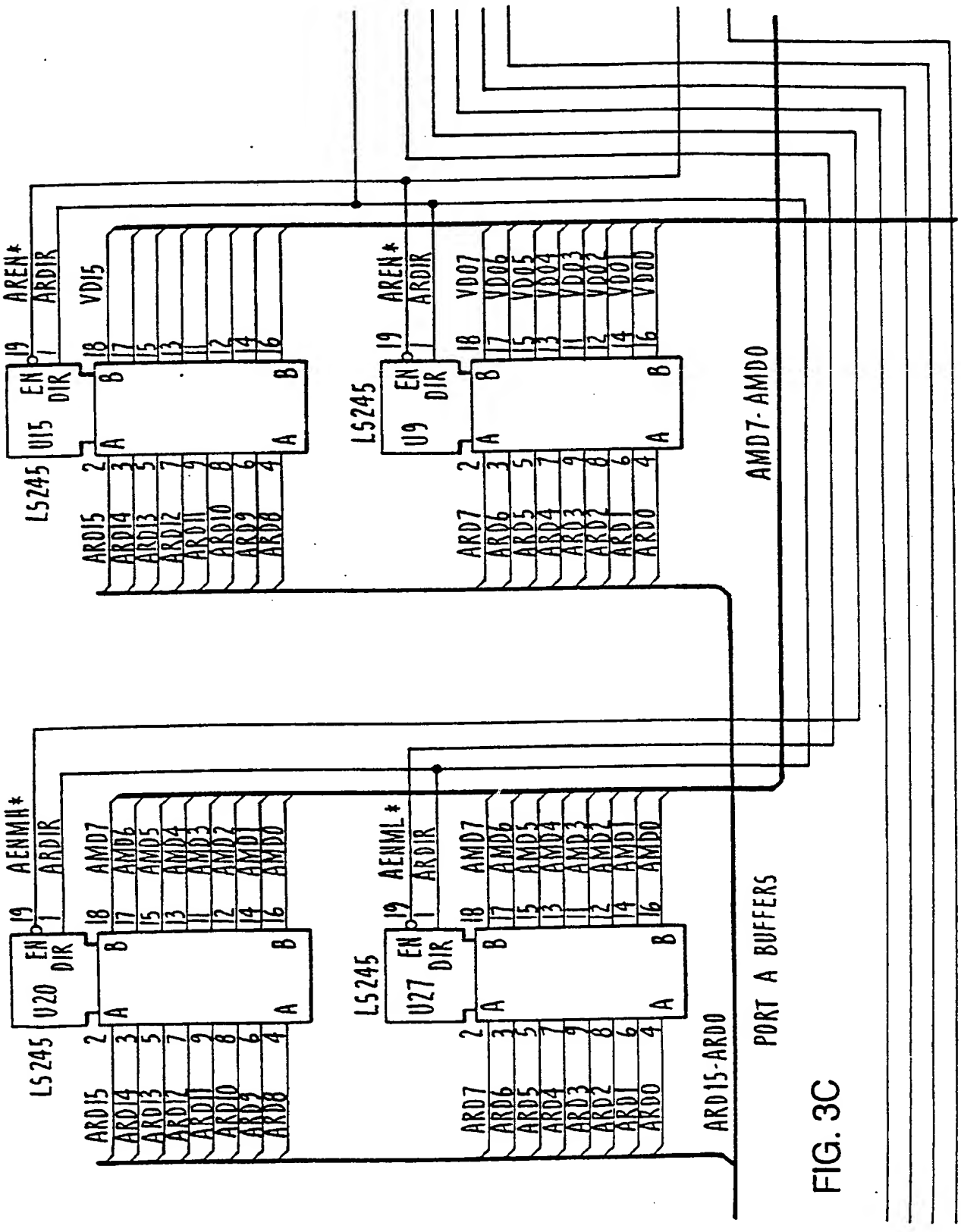

AREN*
ARDIR
VD15
U15
LS245
EN
DIR
A
B
ARD15
ARD14
ARD13
ARD12
ARD11
ARD10
ARD9
ARD8
U9
VD07
VD06
VD05
VD04
VD03
VD02
VD01
VD00
ARD7
ARD6
ARD5
ARD4
ARD3
ARD2
ARD1
ARD0
AMD7-AMD0
AENMH*
AENML*
AMD7
AMD6
AMD5
AMD4
AMD3
AMD2
AMD1
AMD0
U20
U27
ARD15-ARD0
PORT A BUFFERS

FIG. 3C

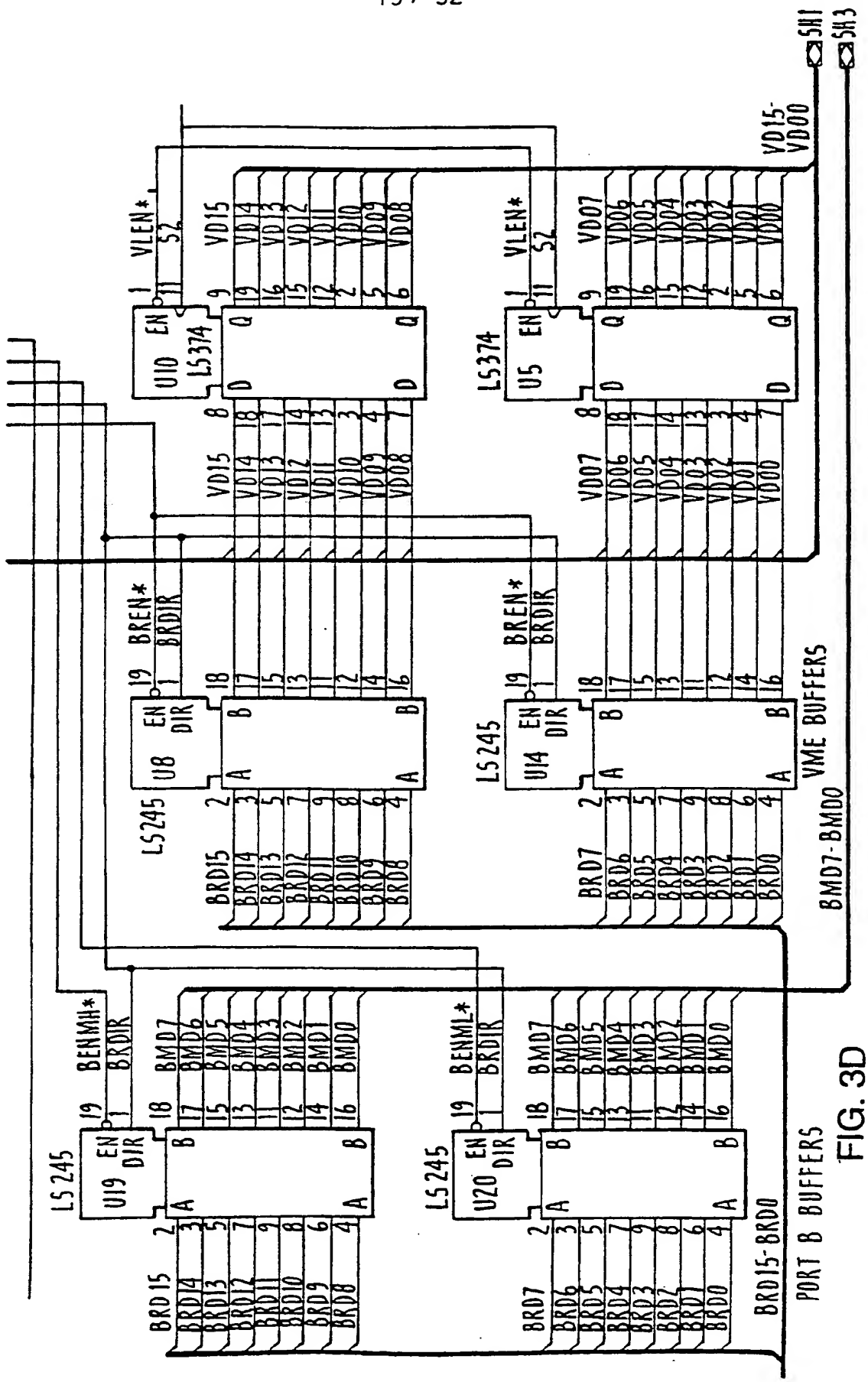

U10 LS374
U5 LS374
U8 LS245
U14 LS245
U19 LS245
U20 LS245
VLEN*
BREN*
BRDIR
BENMH*
BENML*
VD15-VD00
BMD7-BMD0
BRD15-BRD0
VME BUFFERS
PORT B BUFFERS

FIG. 3D

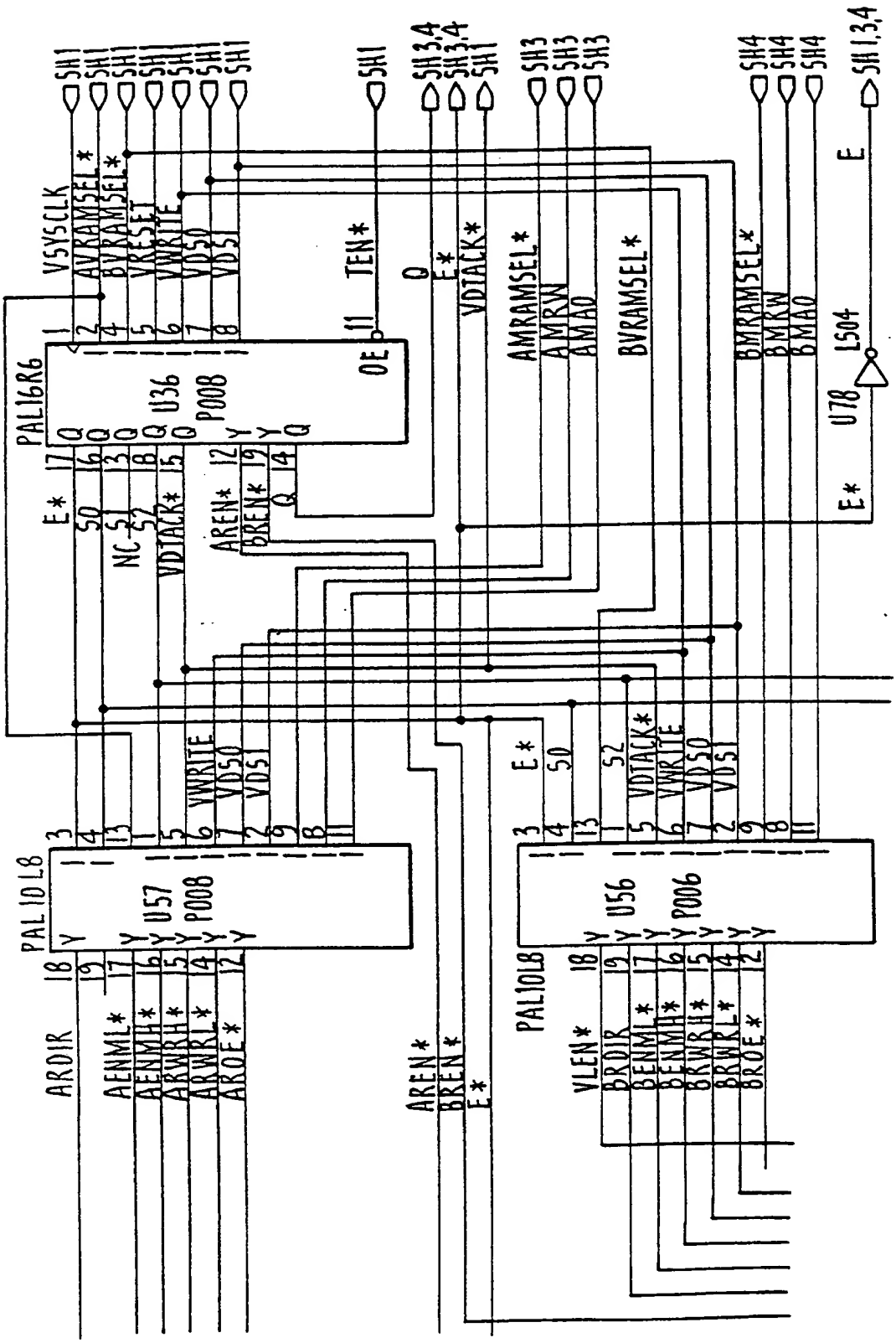


FIG. 3E

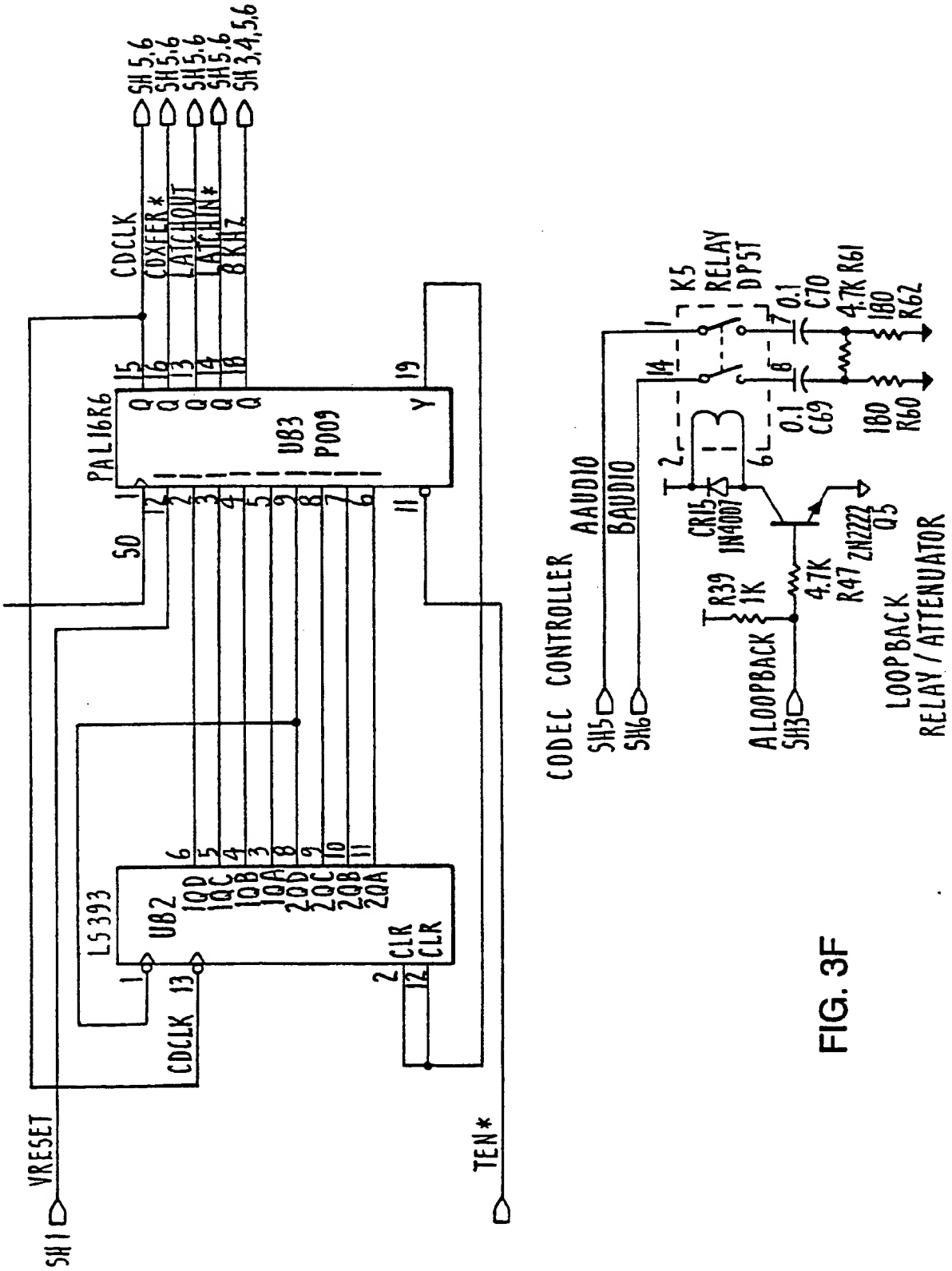
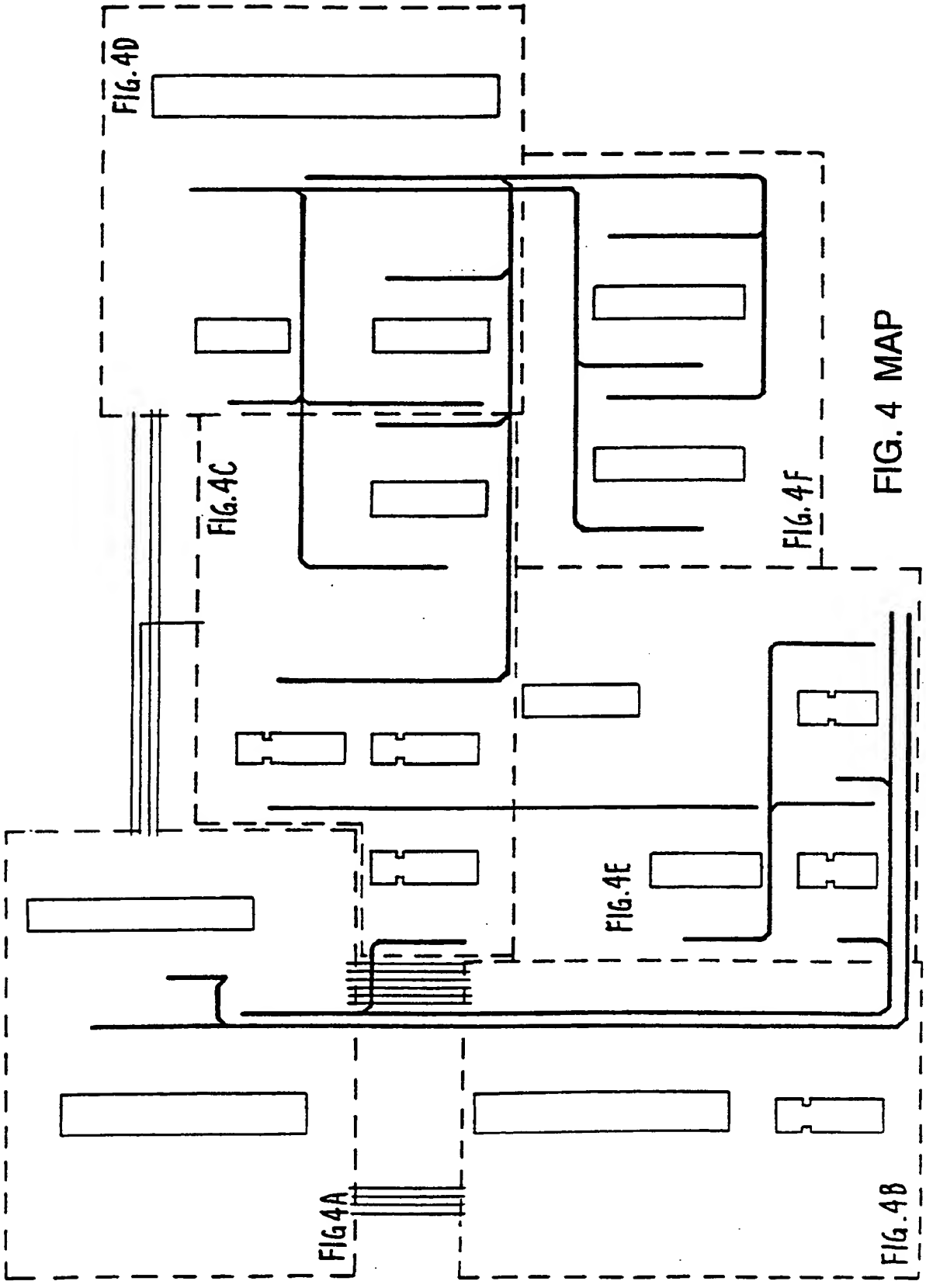

SH 5,6
SH 5,6
SH 5,6
SH 5,6
SH 3,4,5,6
CDCLK
CDXFER*
LATCHOUT
LATCHIN*
8 KHZ
PAL16R6
UB3
P009
50
19
11
LS393
UB2
1QD
1QC
1QB
1QA
2QD
2QC
2QB
2QA
CLR
CDCLK
VRESET
SH 1
TEN*
CODEC CONTROLLER
AAUDIO
BAUDIO
SH5
SH6
K5
RELAY
DPST
0.1
C70
4.7K R61
180
R62
0.1
C69
180
R60
CR15
IN4007
R39
1K
4.7K
R47
ZN2222
Q5
ALOOPBACK
SH3
LOOPBACK
RELAY/ATTENUATOR

FIG. 3F



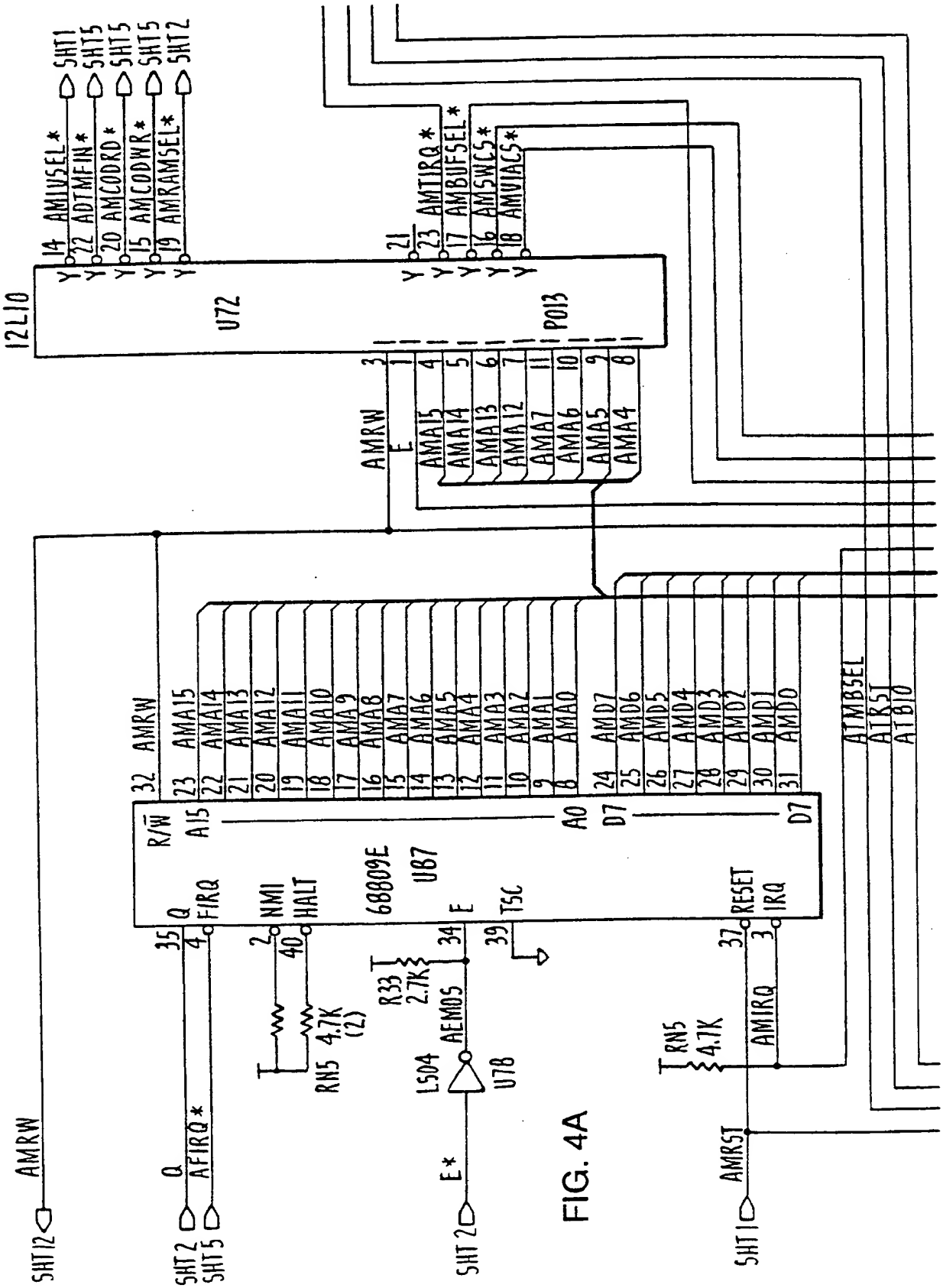

12L10
U72
P013
AMRW
AMA15
AMA14
AMA13
AMA12
AMA7
AMA6
AMA5
AMA4
AMIUSEL*
ADTMFIN*
AMCODRD*
AMCODWR*
AMRAMSEL*
AMTIRQ*
AMBUFSEL*
AMSWCS*
AMVIACS*
SHT1
SHT5
SHT5
SHT5
SHT2
SHT12
SHT2
SHT5
SHT2
SHT1
AMRST
AFIRQ*
AEMOS
AMIRQ
R33 2.7K
RN5 4.7K (2)
RN5 4.7K
L504
U78
U87
68B09E
R/W
A15
A0
D7
FIRQ
NMI
HALT
E
TSC
RESET
IRQ
AMD7
AMD6
AMD5
AMD4
AMD3
AMD2
AMD1
AMD0
ATMBSEL
ATR5I
ATB10

FIG. 4A

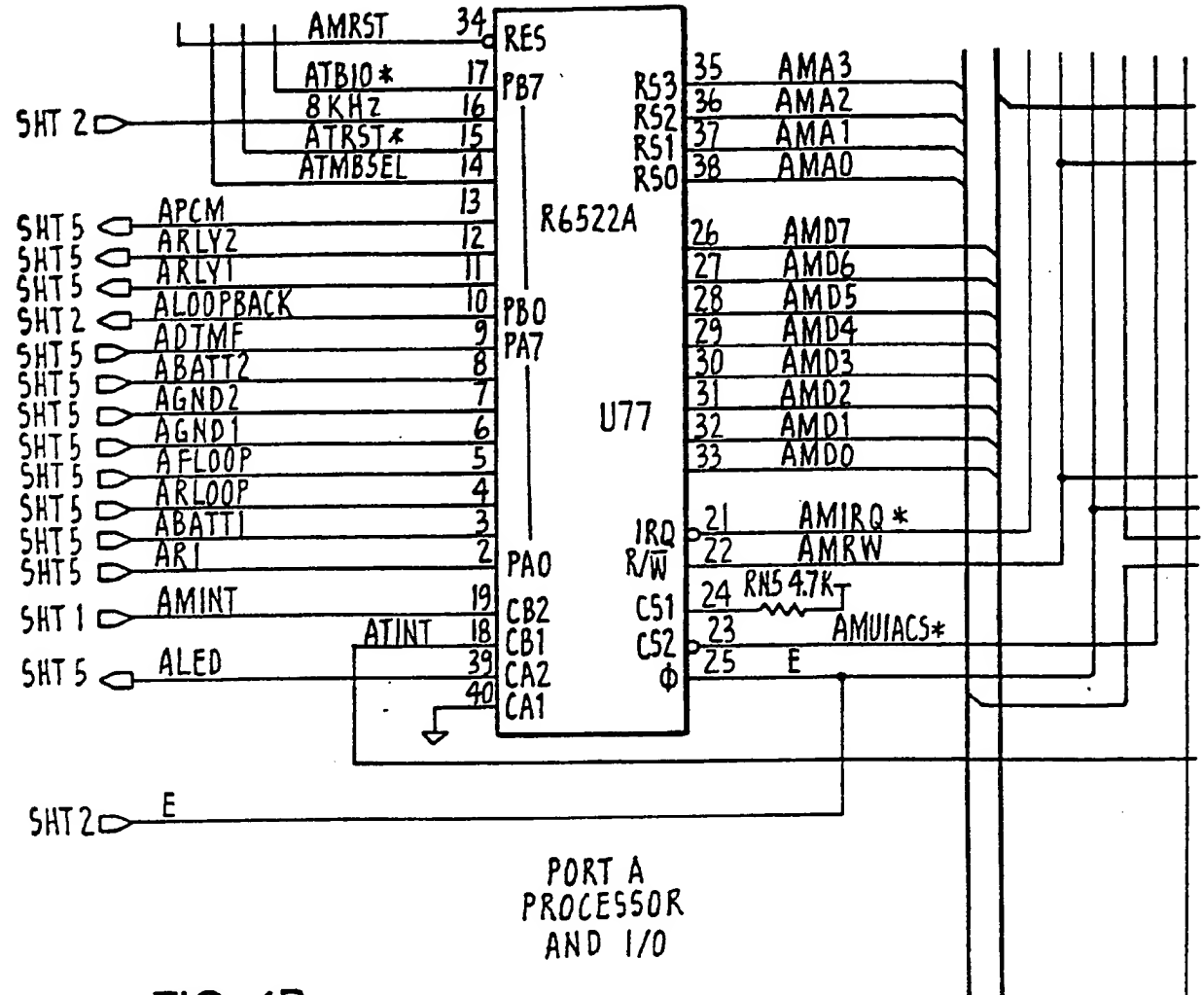
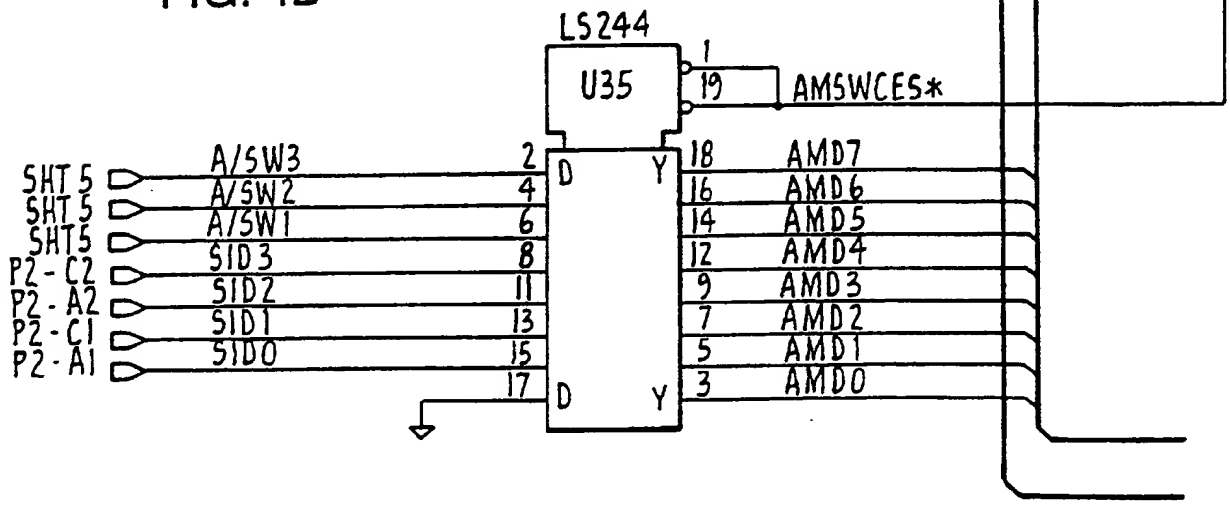


FIG. 4B



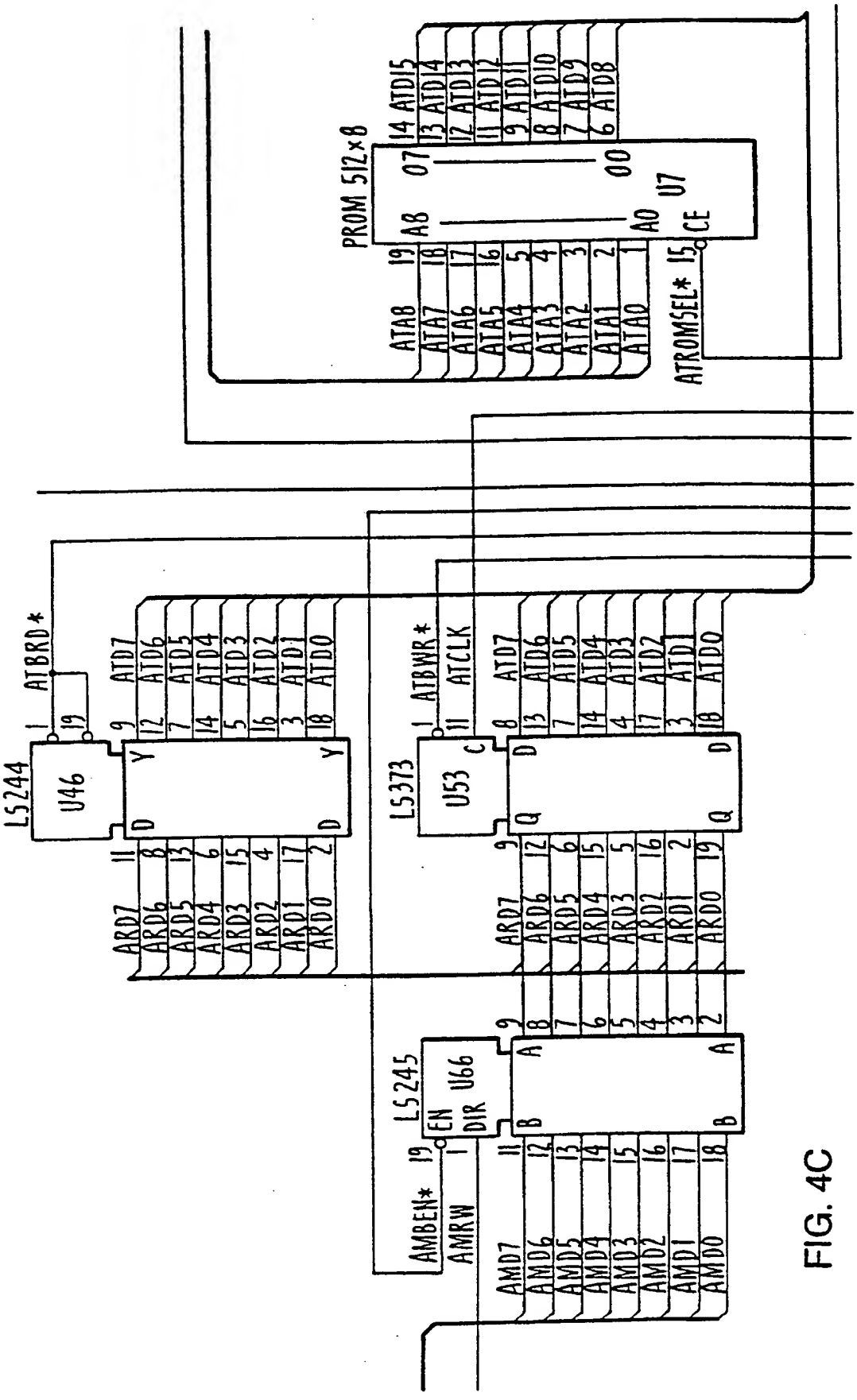

PROM 512x8
U7
ATA8
ATA7
ATA6
ATA5
ATA4
ATA3
ATA2
ATA1
ATA0
ATROMSEL* 15
ATD15
ATD14
ATD13
ATD12
ATD11
ATD10
ATD9
ATD8
LS244
U46
ATBRD*
ARD7
ARD6
ARD5
ARD4
ARD3
ARD2
ARD1
ARD0
ATD7
ATD6
ATD5
ATD4
ATD3
ATD2
ATD1
ATD0
LS373
U53
ATBWR*
ATCLK
LS245
U66
AMBEN*
AMRW
AMD7
AMD6
AMD5
AMD4
AMD3
AMD2
AMD1
AMD0

FIG. 4C

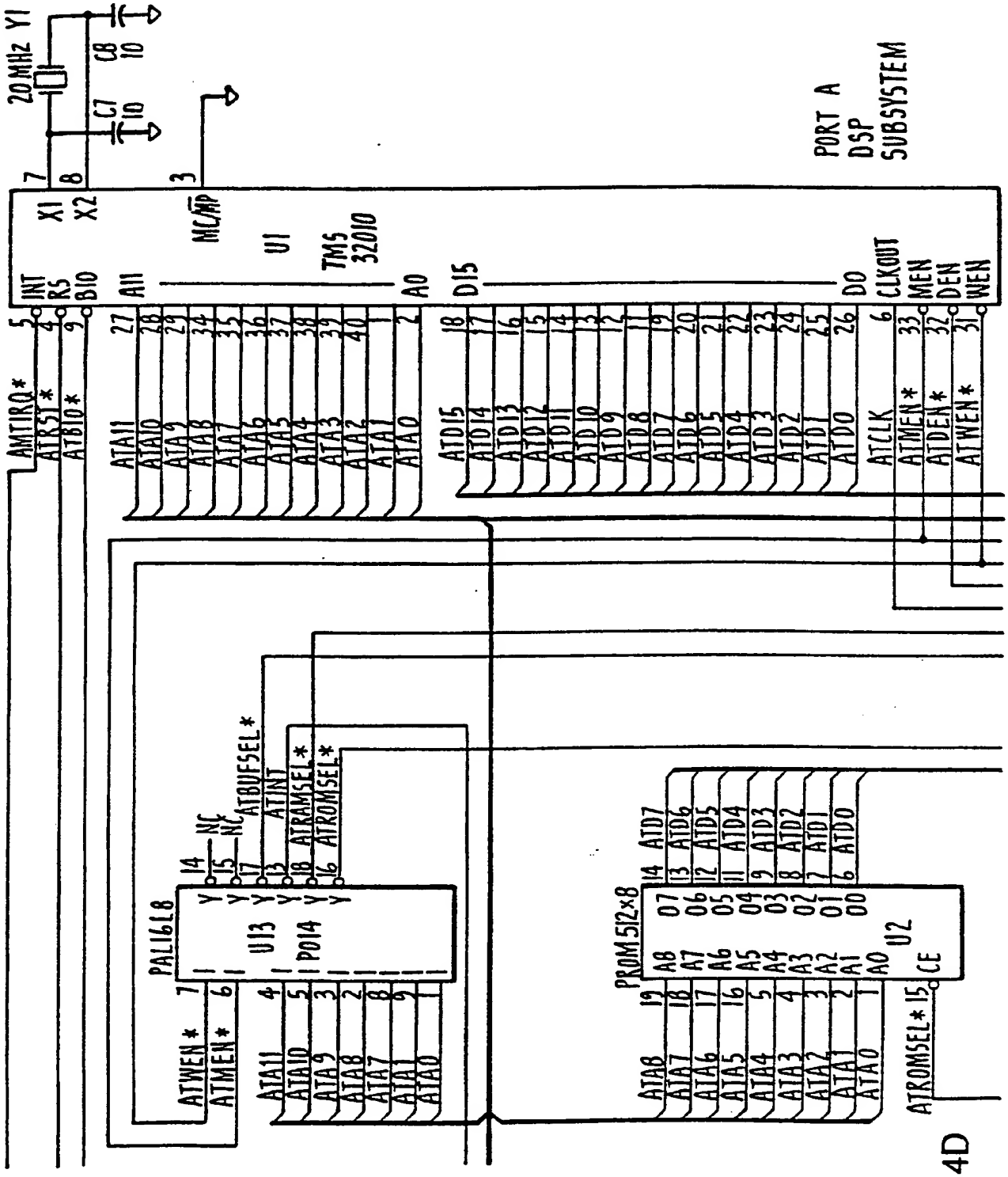

PORT A
DSP
SUBSYSTEM
20 MHz Y1
C8 10
C7 10
X1
X2
MC/MP
U1
TMS 32010
INT
R5
B10
A11
A0
D15
D0
CLKOUT
MEN
DEN
WEN
AMTIRQ*
ATR5I*
ATB10*
ATA11
ATA10
ATA9
ATA8
ATA7
ATA6
ATA5
ATA4
ATA3
ATA2
ATA1
ATA0
ATD15
ATD14
ATD13
ATD12
ATD11
ATD10
ATD9
ATD8
ATD7
ATD6
ATD5
ATD4
ATD3
ATD2
ATD1
ATD0
ATCLK
ATMEN*
ATDEN*
ATWEN*
ATBUFSEL*
ATINT
ATRAMSEL*
ATROMSEL*
NC
PAL16L8
U13
P014
PROM 512x8
U2
CE
ATWEN*
ATMEN*
ATROMSEL*15

FIG. 4D

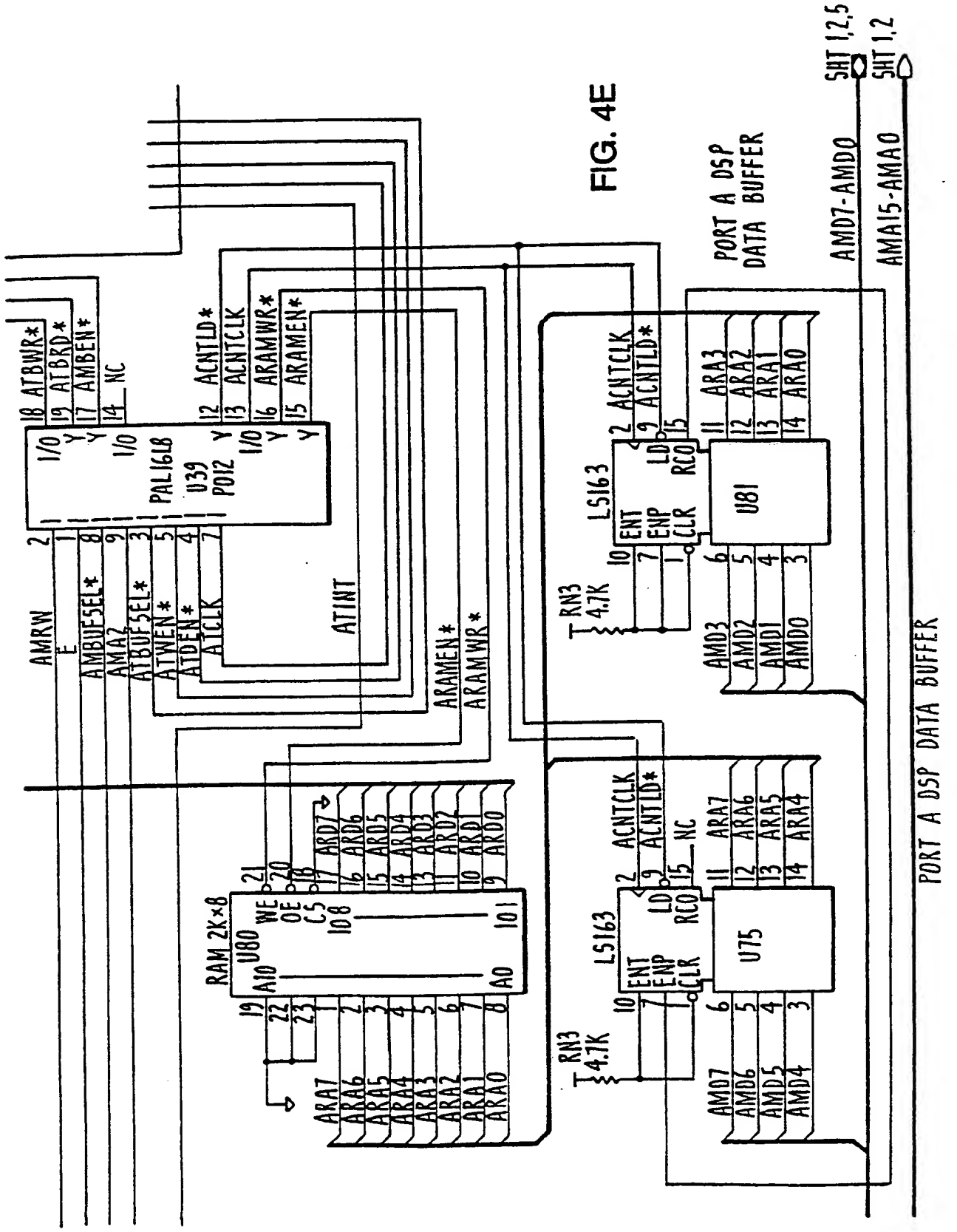

SHT 1,2,5
SHT 1,2
AMD7-AMD0
AMA15-AMA0
PORT A DSP DATA BUFFER
PORT A DSP DATA BUFFER
U81
U75
U80
LS163
RAM 2K×8
PAL16L8
U39
P012
RN3
4.7K
ATINT
ARAMEN*
ARAMWR*
ACNTLD*
ACNTCLK
AMRW
AMBUFSEL*
AMA2
ATBUFSEL*
ATWEN*
ATDEN*
ATCLK
ATBWR*
ATBRD*
AMBEN*
NC

FIG. 4E

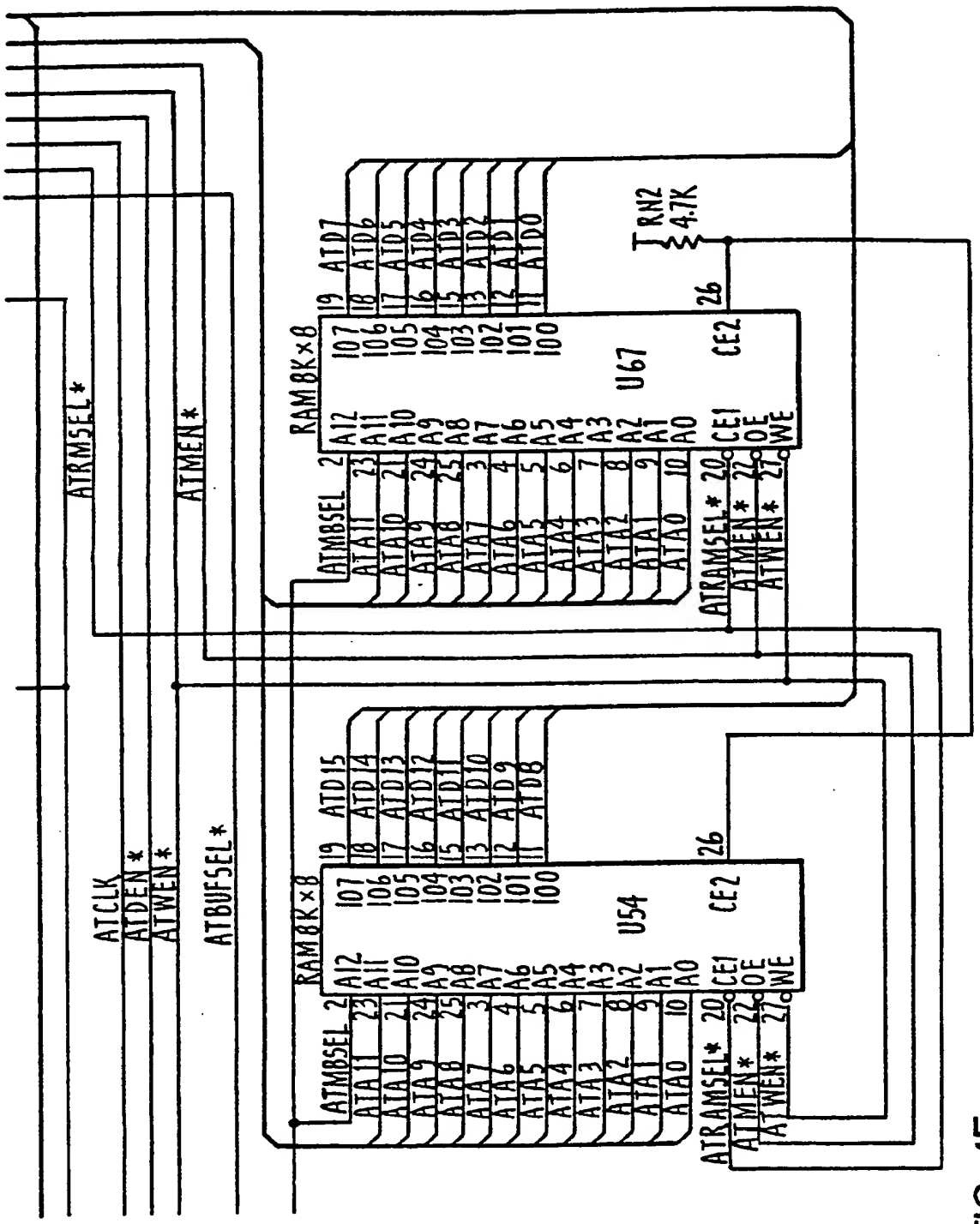


FIG. 4F

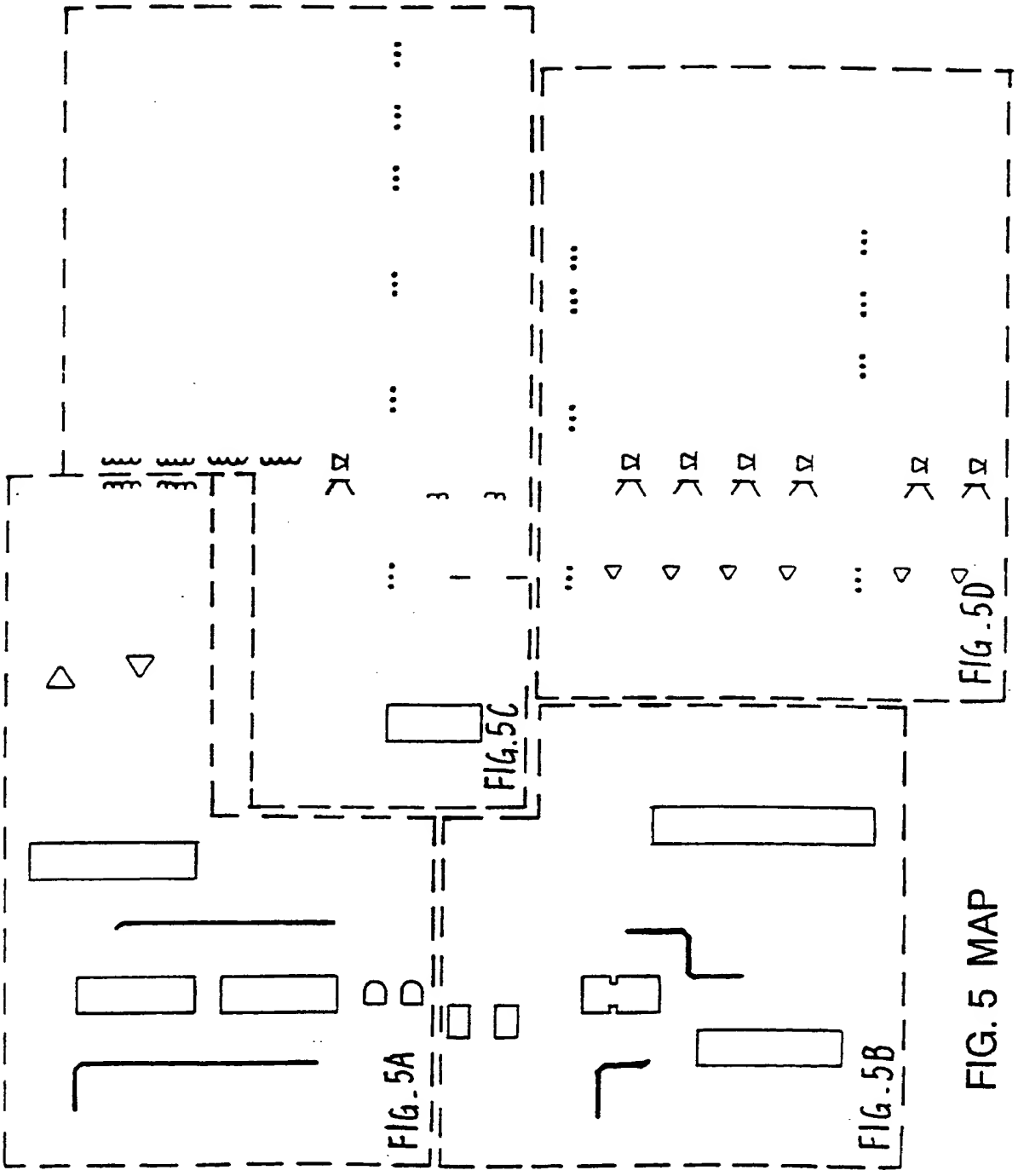


FIG. 5 MAP

SUBSTITUTE SHEET

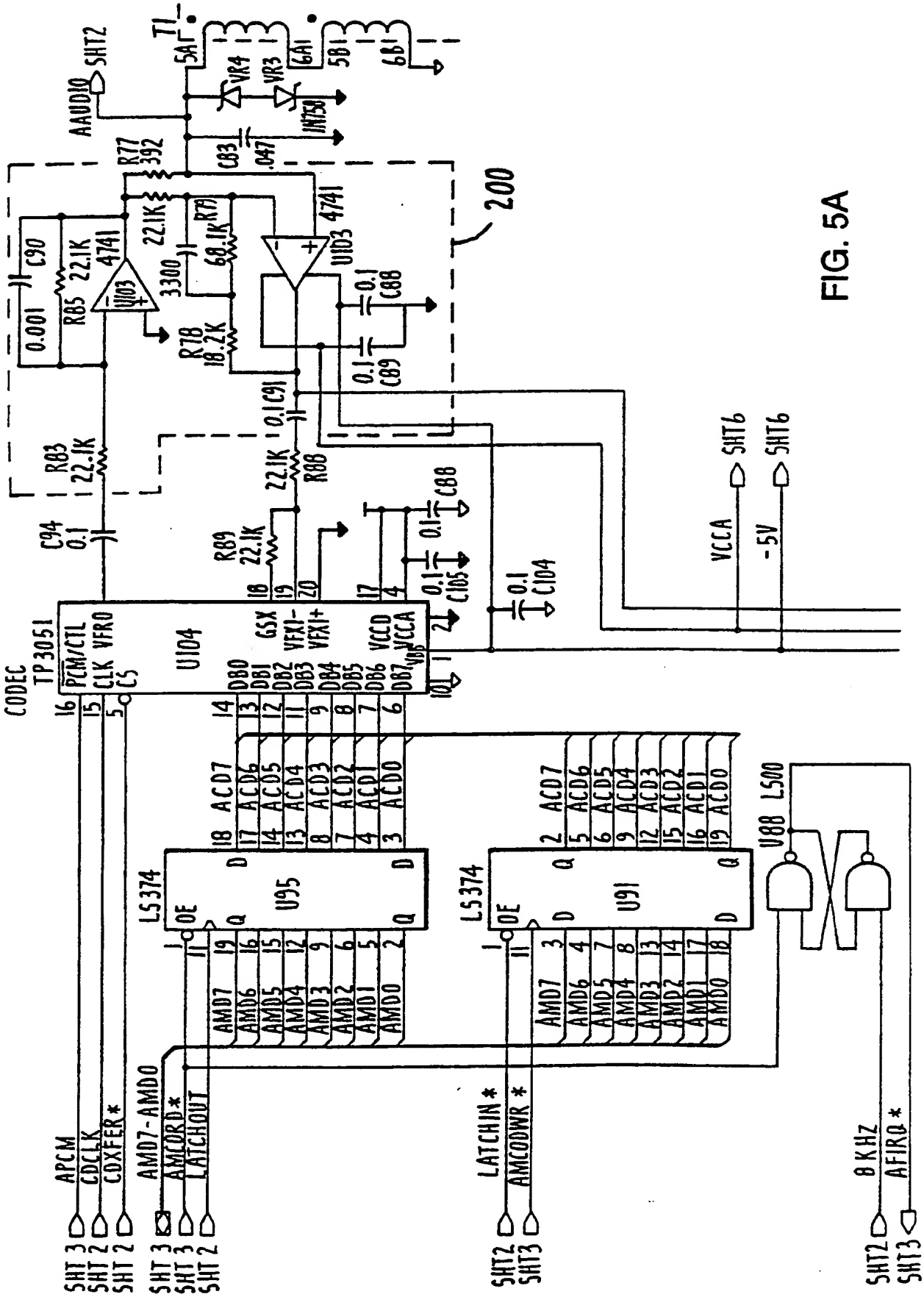

200
CODEC
TP3051
U104
PCM/CTL
CLK
CS
VFRO
GSX
VFXI-
VFXI+
VCCD
VCCA
VBB
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
R77 392
C83 .047
R85 22.1K
R83 22.1K
R88 22.1K
R89 22.1K
R78 18.2K
R79 68.1K
3300
C90 0.001
C94 0.1
C88 0.1
C89 0.1
C91 0.1
C105 0.1
C104 0.1
U103 4741
U103 4741
VR4
VR3
IN750
AAUDIO
SHT2
T1
5A
6A1
5B1
6B1
VCCA
-5V
SHT6
SHT6
U95
U91
LS374
LS374
OE
D
Q
ACD7
ACD6
ACD5
ACD4
ACD3
ACD2
ACD1
ACD0
AMD7
AMD6
AMD5
AMD4
AMD3
AMD2
AMD1
AMD0
U88 LS00
APCM
CDCLK
CDXFER*
AMD7-AMD0
AMCORD*
LATCHOUT
LATCHIN*
AMCOWR*
8 KHZ
AFIRQ*
SHT3
SHT2
SHT3
SHT2
SHT3
SHT2
SHT3
SHT2
SHT3
SHT2

FIG. 5A

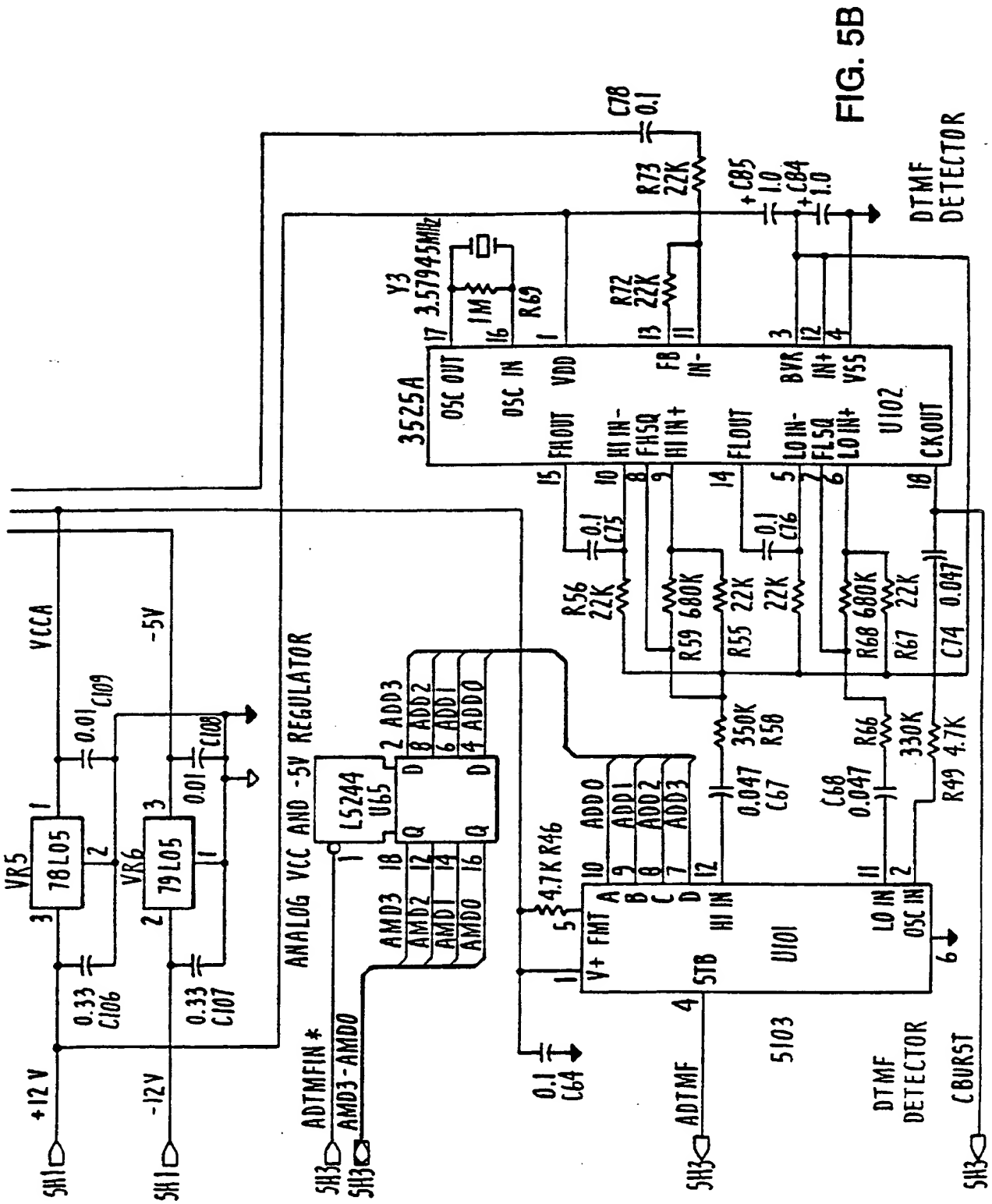

VCCA
-5V
+12 V
-12V
VR5
78L05
VR6
79L05
0.01 C109
0.01 C108
0.33 C106
0.33 C107
ANALOG VCC AND -5V REGULATOR
ADTMFIN*
AMD3-AMD0
L5244 U65
ADD3
ADD2
ADD1
ADD0
AMD3
AMD2
AMD1
AMD0
4.7K R46
0.1 C64
U101
V+ FMT
STB
HI IN
LO IN
OSC IN
5103
ADTMF
DTMF DETECTOR
CBURST
3525A
OSC OUT
OSC IN
VDD
FHOUT
HI IN-
HI IN+
FLOUT
LO IN-
LO IN+
CKOUT
FB
IN-
BYR
IN+
VSS
U102
Y3 3.57945MHz
1M R69
R72 22K
R73 22K
C78 0.1
C85 1.0
C84 1.0
0.1 C75
0.1 C76
R56 22K
R59 680K
R55 22K
22K
R68 680K
R67 22K
C74 0.047
350K R58
0.047 C67
R66 330K
C68 0.047
R49 4.7K
DTMF DETECTOR

FIG. 5B

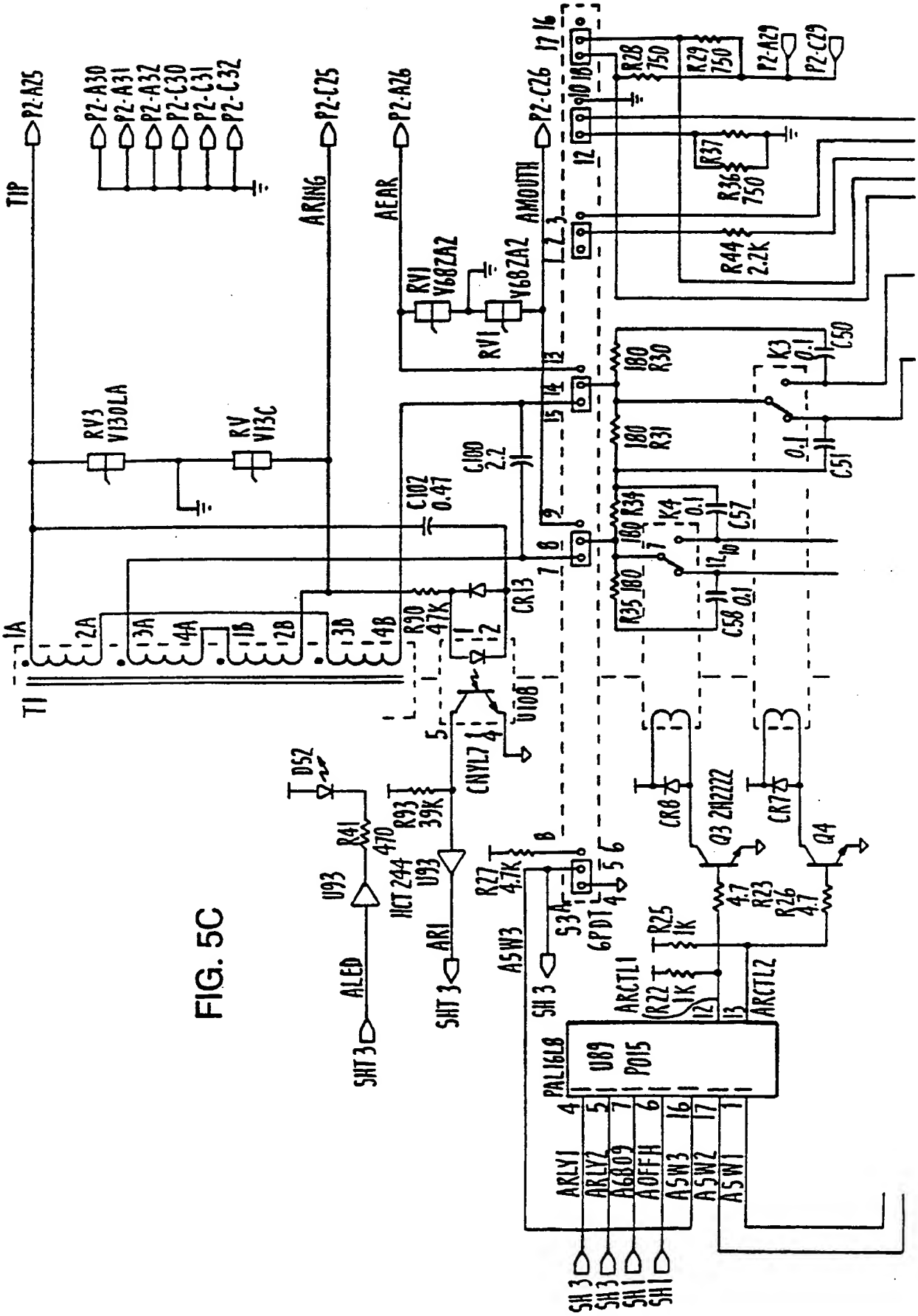


FIG. 5C

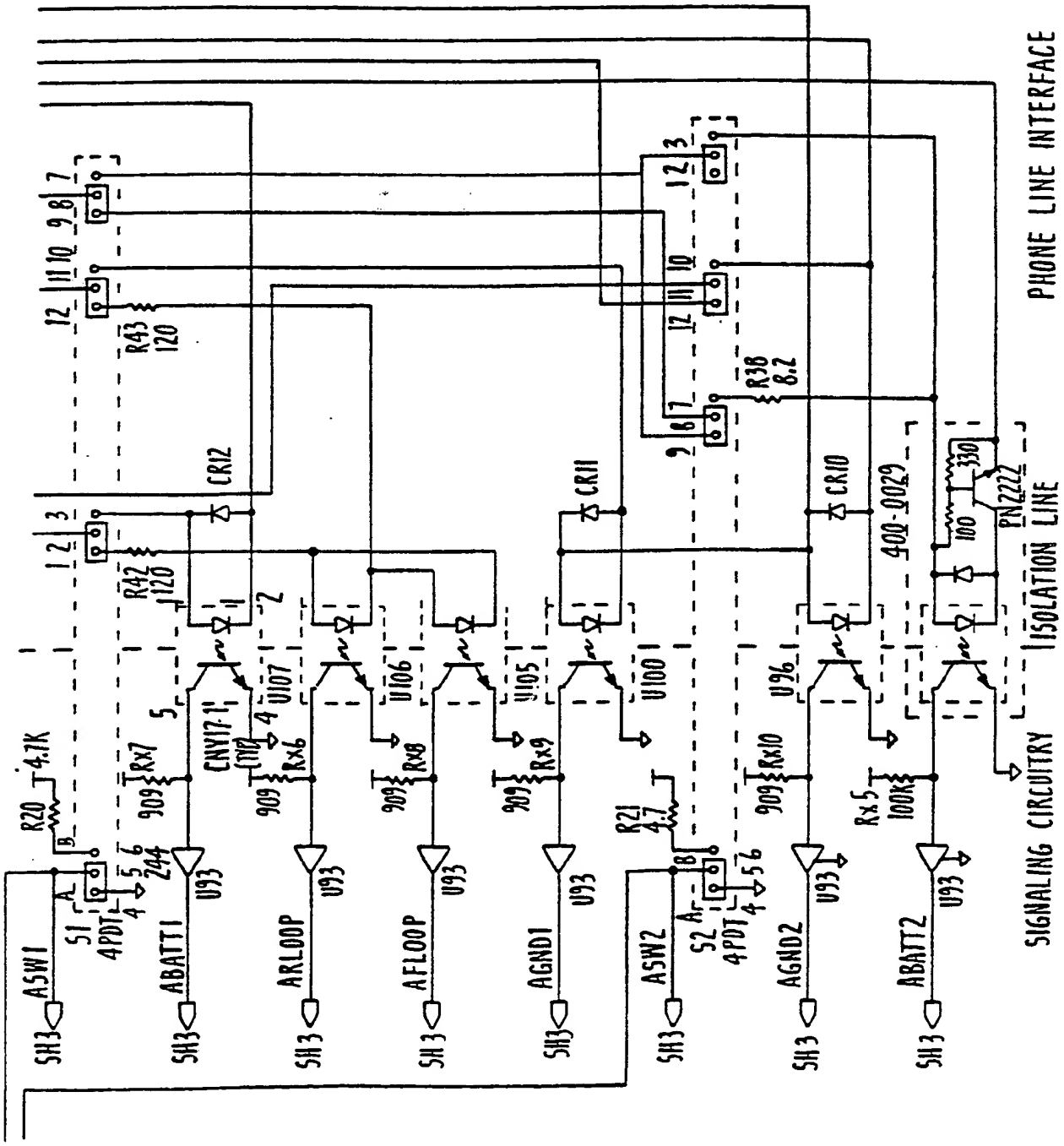

PHONE LINE INTERFACE
ISOLATION LINE
SIGNALING CIRCUITRY
ASW1
ABATT1
ARLOOP
AFLOOP
AGND1
ASW2
AGND2
ABATT2
SH3
U93
R20 4.7K
R21 4.7
R42 120
R43 120
R38 8.2
R44 100
R45 100K
RX5
RX6
RX7
RX8
RX9
RX10
909
244
330
CR10
CR11
CR12
400-0029
PN2222
U96
U100
U105
U106
U107
CNY17-1
4PDT

FIG. 5D

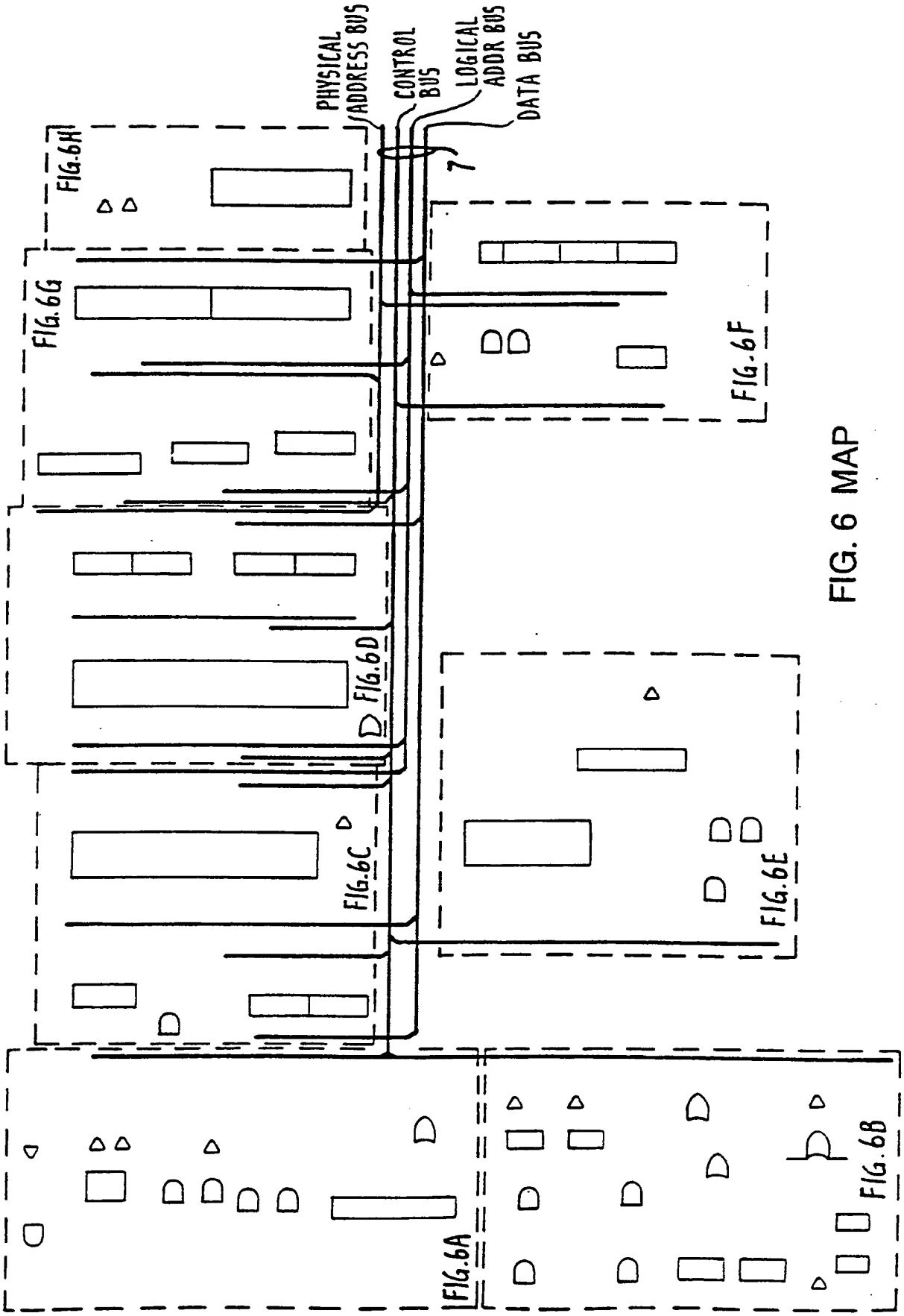


FIG. 6 MAP

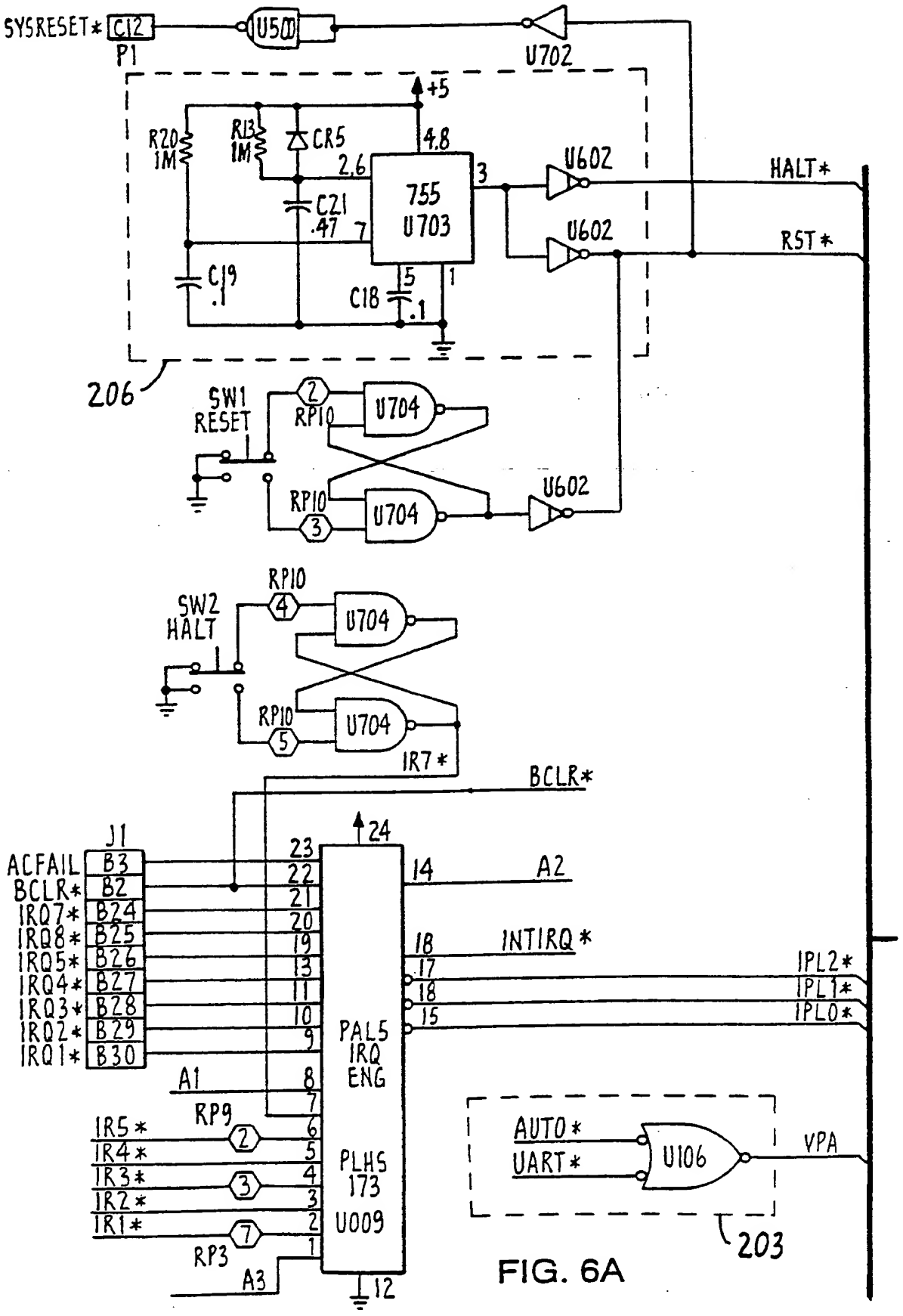

SYSRESET*
C12
P1
U500
U702
+5
R20
1M
R13
1M
CR5
4.8
2.6
755
U703
3
U602
HALT*
C21
.47
7
U602
R5T*
C19
.1
15
1
C18
.1
206
SW1
RESET
2
RP10
U704
RP10
3
U704
U602
RP10
SW2
HALT
4
U704
RP10
5
U704
IR7*
BCLR*
J1
ACFAIL B3
BCLR* B2
IRQ7* B24
IRQ8* B25
IRQ5* B26
IRQ4* B27
IRQ3* B28
IRQ2* B29
IRQ1* B30
24
23
22
21
20
19
13
11
10
9
14
A2
18
INTIRQ*
17
18
15
IPL2*
IPL1*
IPL0*
PAL5
IRQ
ENG
A1
8
7
RP9
IR5*
IR4*
IR3*
IR2*
IR1*
2
3
7
6
5
4
3
2
1
PLHS
173
U009
RP3
A3
12
AUTO*
UART*
U106
VPA
203

FIG. 6A

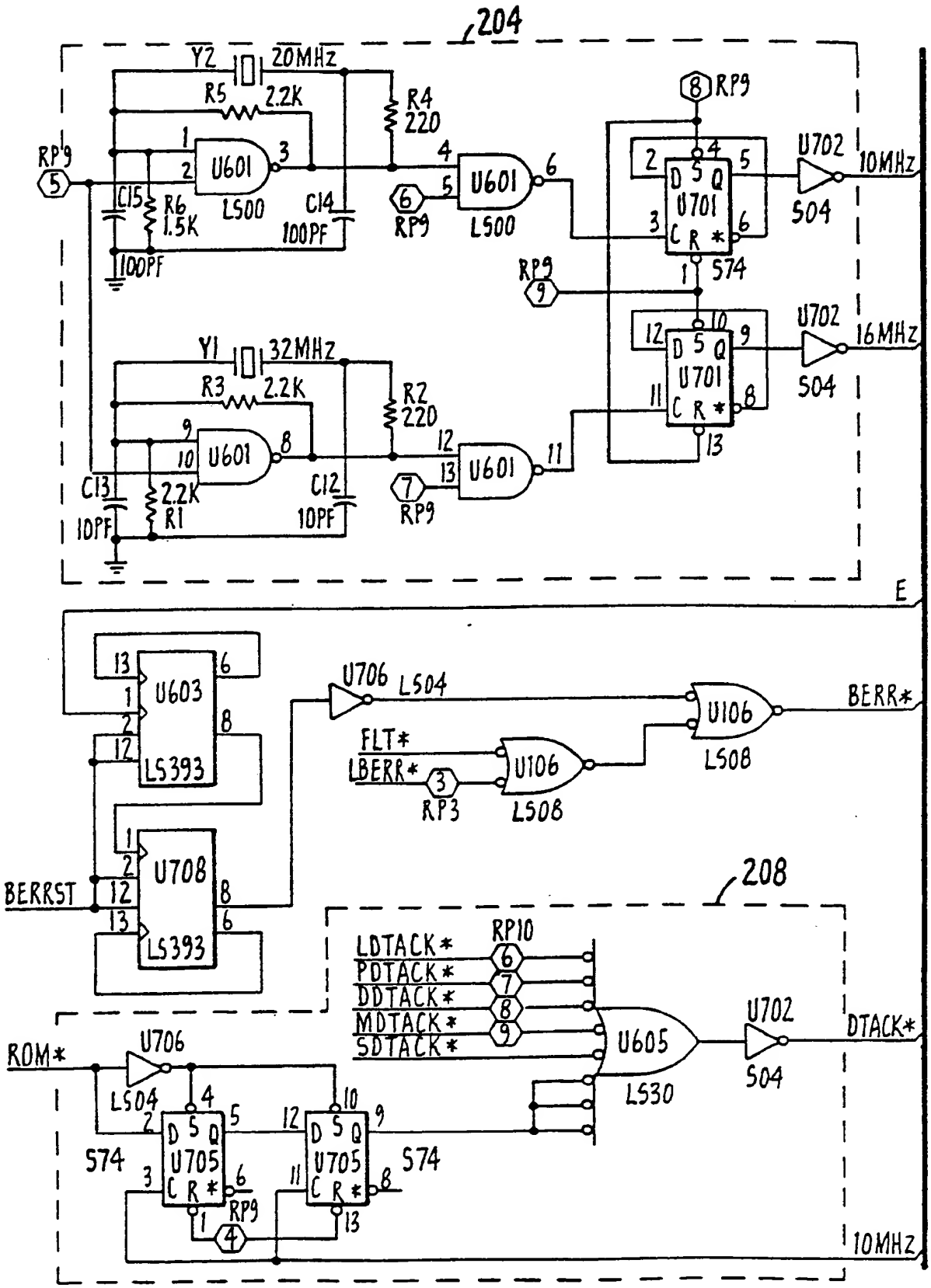


FIG. 6B

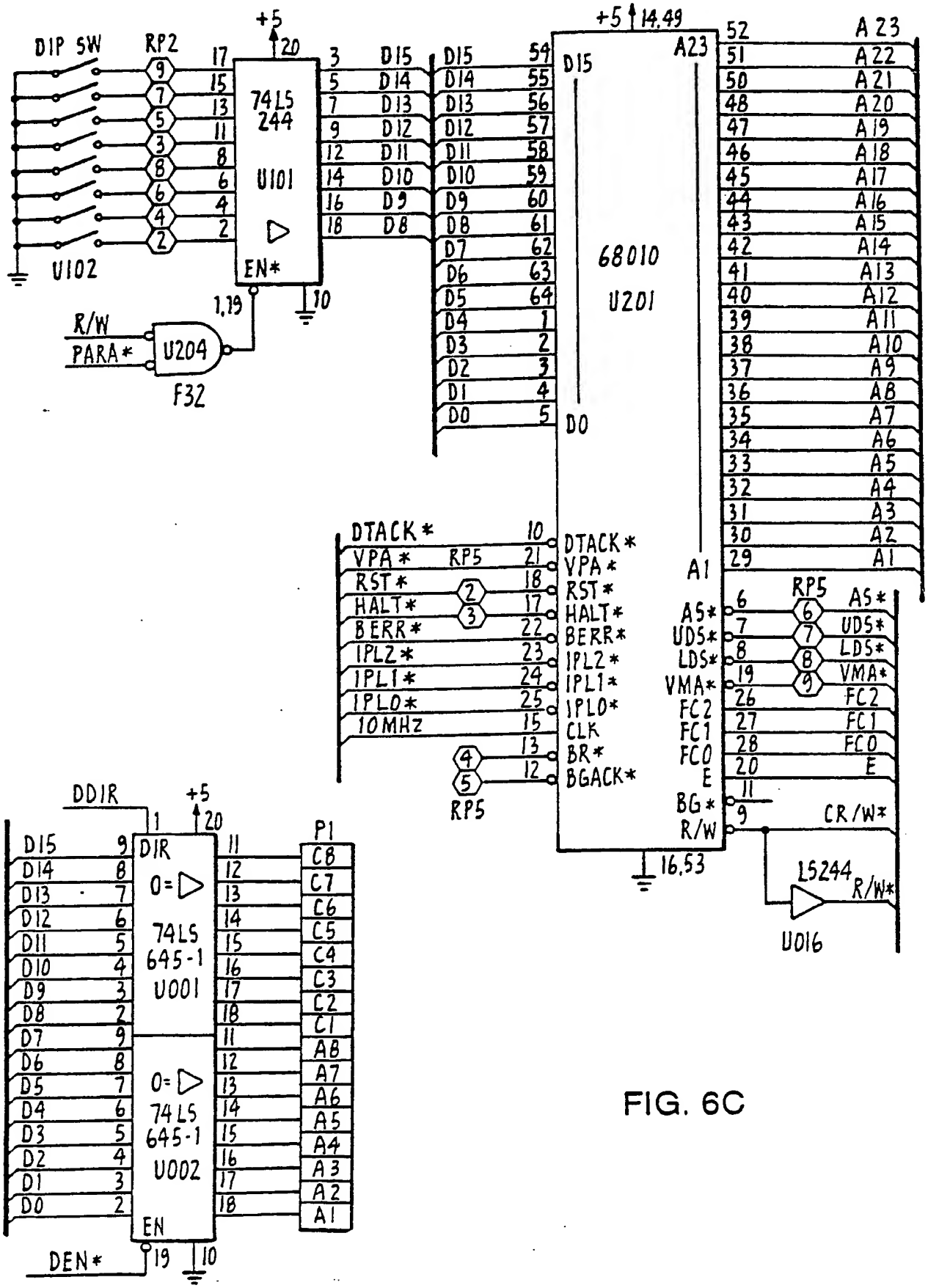


FIG. 6C

SUBSTITUTE SHEET

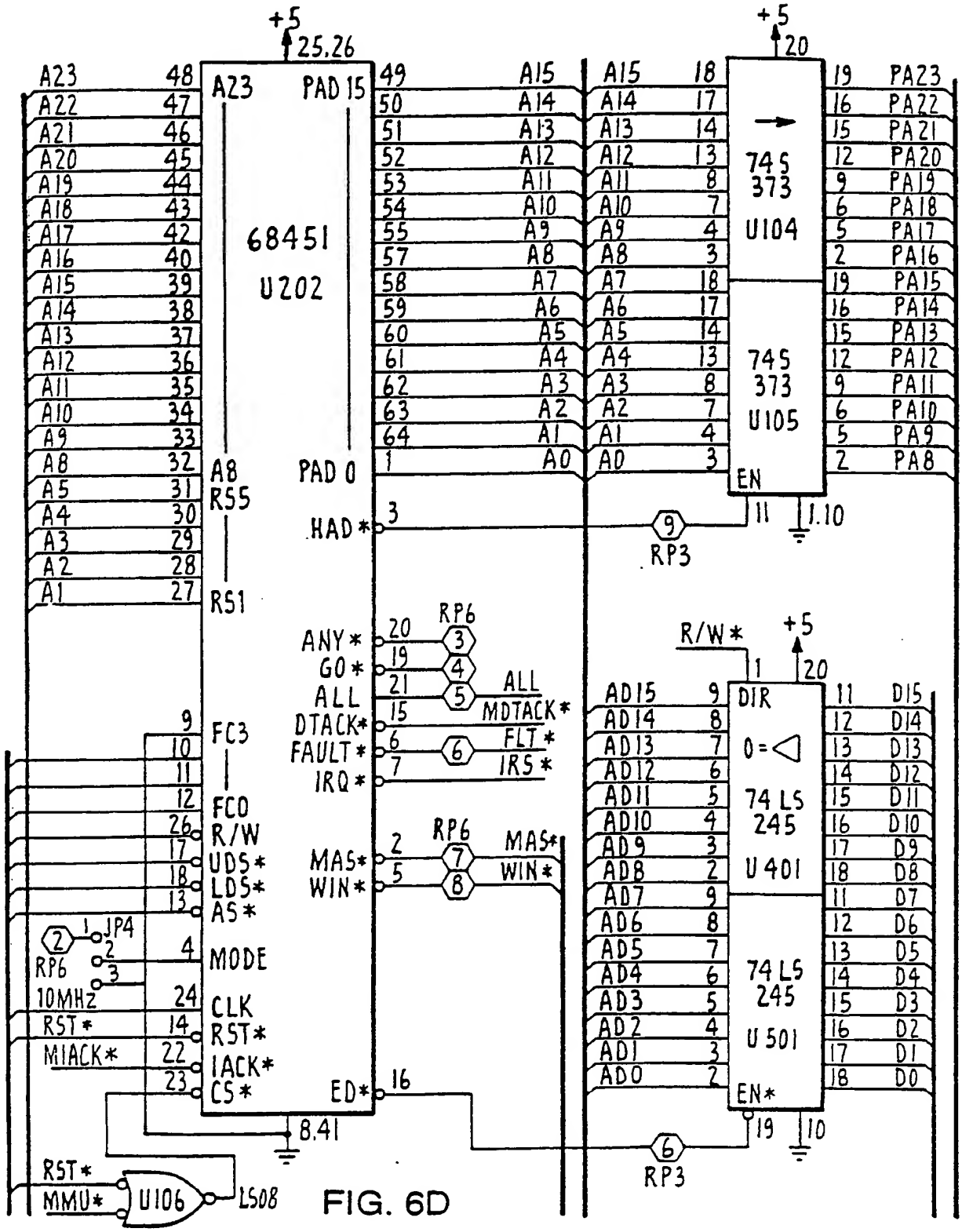


FIG. 6D

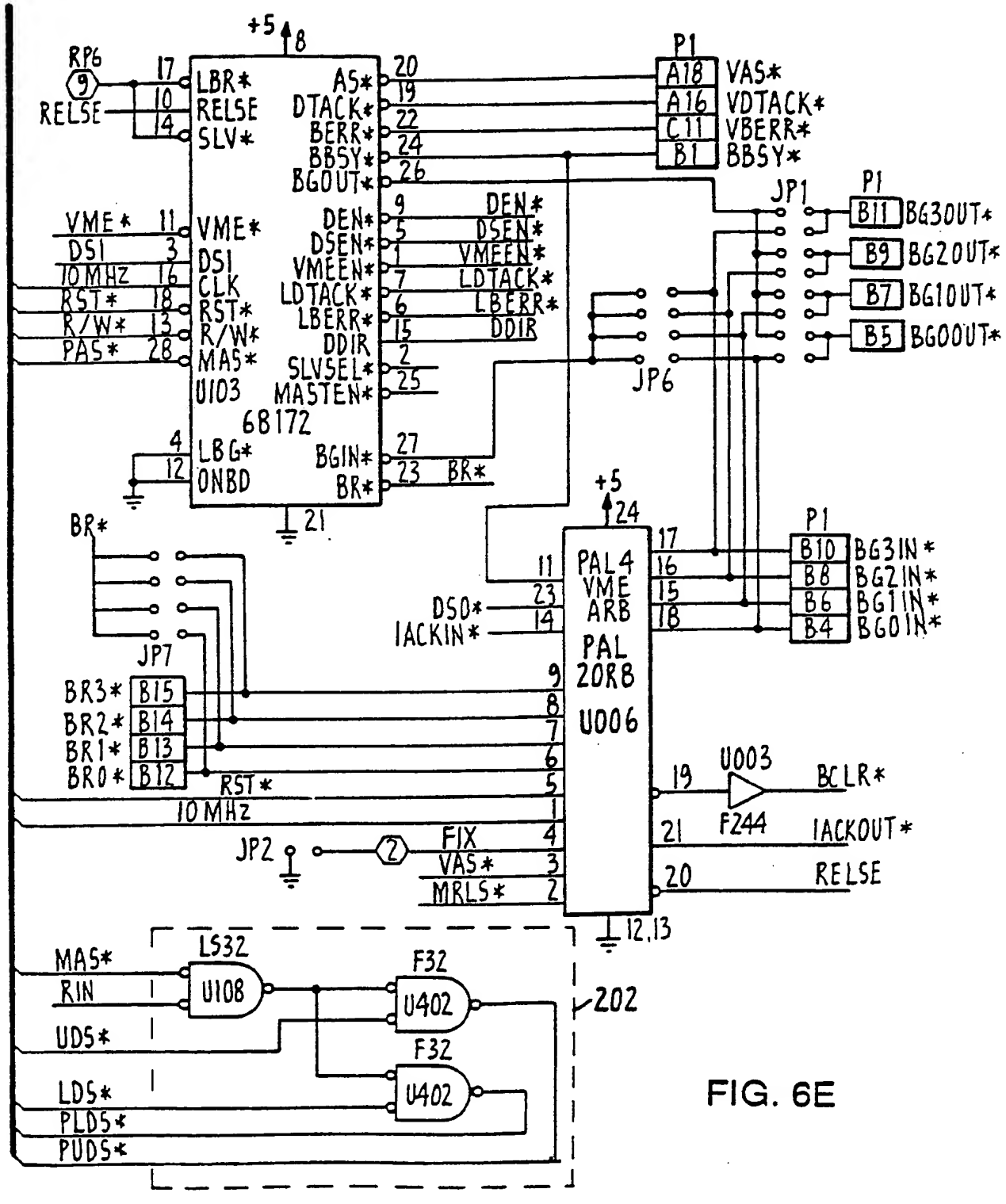

+5
8
RP6
9
RELSE
17
10
14
LBR*
RELSE
SLV*
AS*
DTACK*
BERR*
BBSY*
BGOUT*
20
19
22
24
26
P1
A18 VAS*
A16 VDTACK*
C11 VBERR*
B1 BBSY*
VME* 11
DS1 3
10MHz 16
RST* 18
R/W* 13
PAS* 28
VME*
DS1
CLK
RST*
R/W*
MAS*
U103
68172
DEN*
DSEN*
VMEEN*
LDTACK*
LBERR*
DDIR
SLVSEL*
MASTEN*
9
5
1
7
6
15
2
25
DEN*
DSEN*
VMEEN*
LDTACK*
LBERR*
DDIR
JP1
P1
B11 BG3OUT*
B9 BG2OUT*
B7 BG1OUT*
B5 BG0OUT*
JP6
4
12
LBG*
ONBD
BGIN*
BR*
27
23
BR*
21
+5
24
BR*
JP7
BR3* B15
BR2* B14
BR1* B13
BR0* B12
RST*
10MHz
JP2
2
FIX
VAS*
MRLS*
4
3
2
11
23
14
DS0*
IACKIN*
PAL4
VME
ARB
PAL
20R8
U006
9
8
7
6
5
1
17
16
15
18
P1
B10 BG3IN*
B8 BG2IN*
B6 BG1IN*
B4 BG0IN*
U003
F244
19
21
20
BCLR*
IACKOUT*
RELSE
12,13
202
MAS*
RIN
UDS*
LDS*
PLDS*
PUDS*
LS32
U108
F32
U402
F32
U402

FIG. 6E

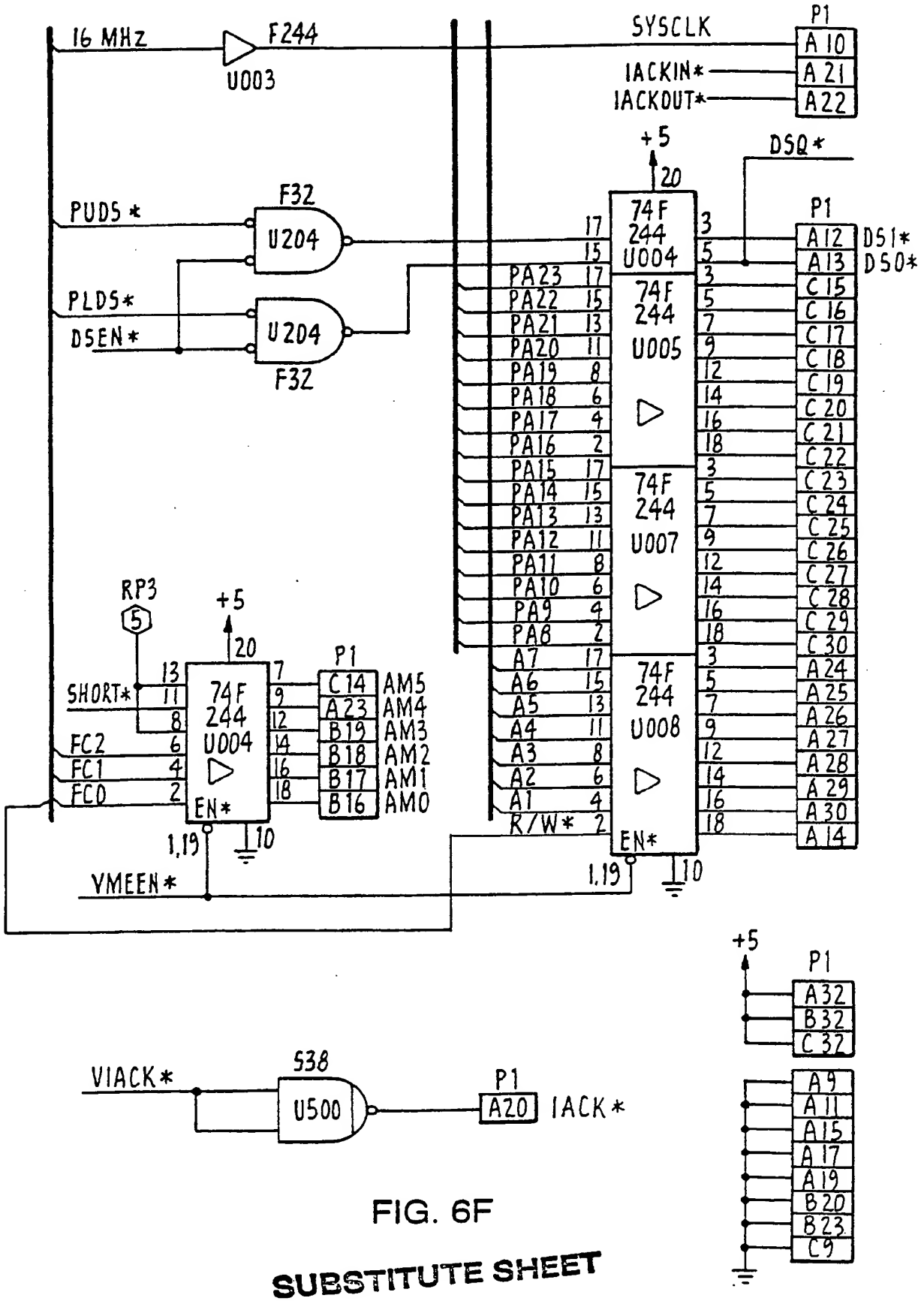


FIG. 6F

SUBSTITUTE SHEET

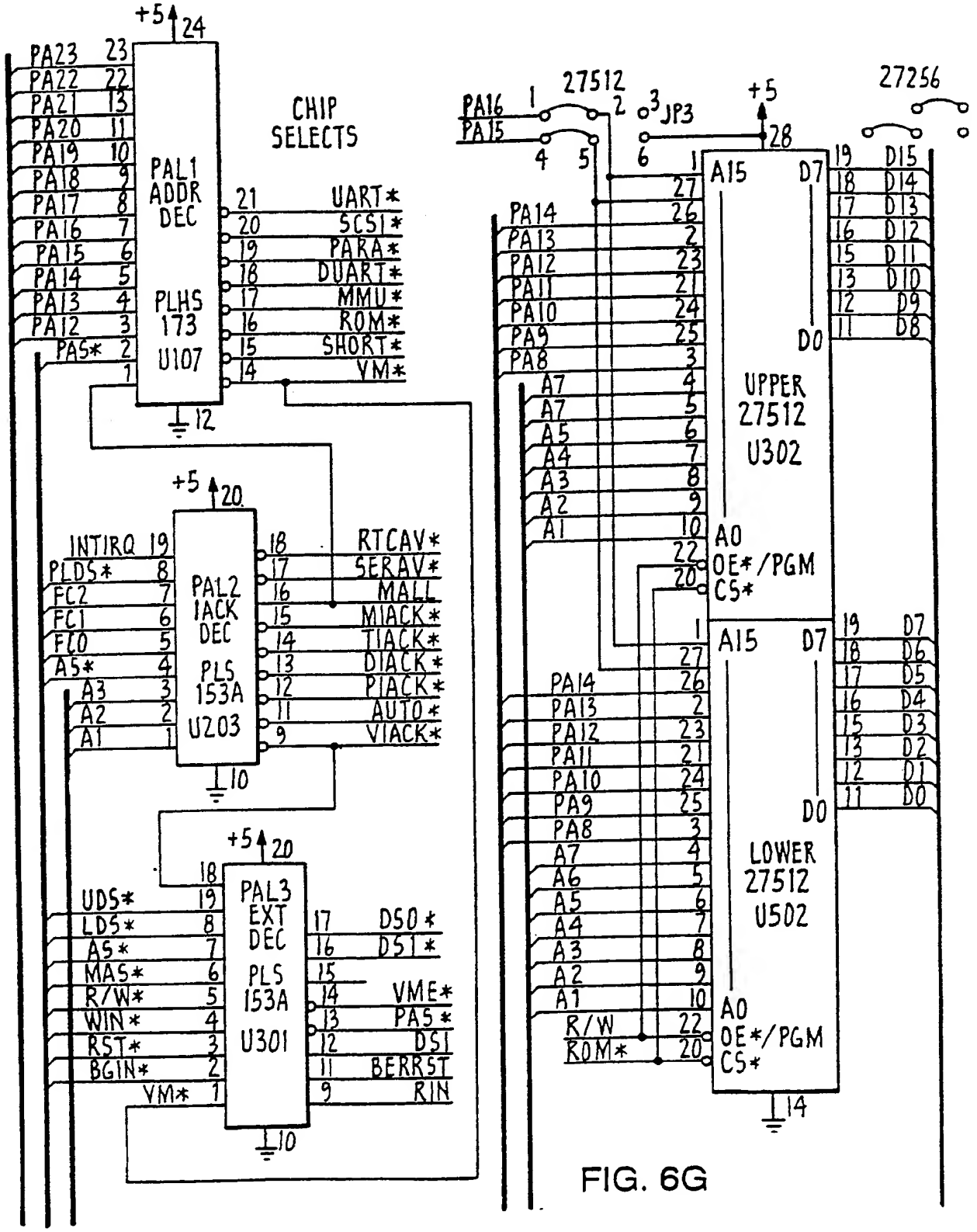
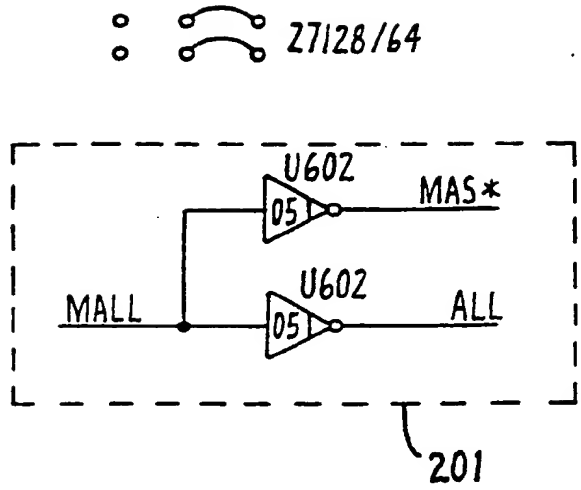

CHIP SELECTS
PAL1 ADDR DEC
PLHS 173 U107
PAL2 IACK DEC
PLS 153A U203
PAL3 EXT DEC
PLS 153A U301
27512
27256
JP3
UPPER 27512 U302
LOWER 27512 U502
UART*
SCSI*
PARA*
DUART*
MMU*
ROM*
SHORT*
VM*
RTCAV*
SERAV*
MALL
MIACK*
TIACK*
DIACK*
PIACK*
AUTO*
VIACK*
DS0*
DS1*
VME*
PAS*
DSI
BERRST
RIN
A15
A0
D7
D0
OE*/PGM
CS*
R/W
ROM*

FIG. 6G



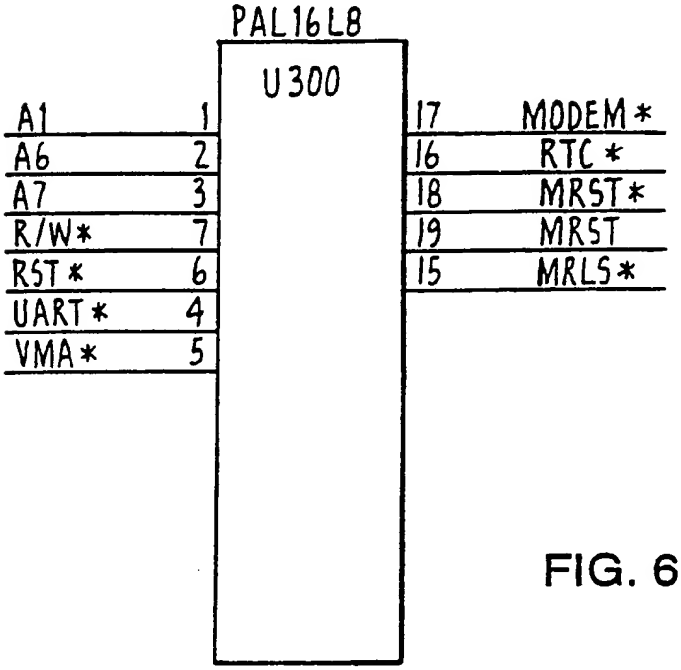


FIG. 6H

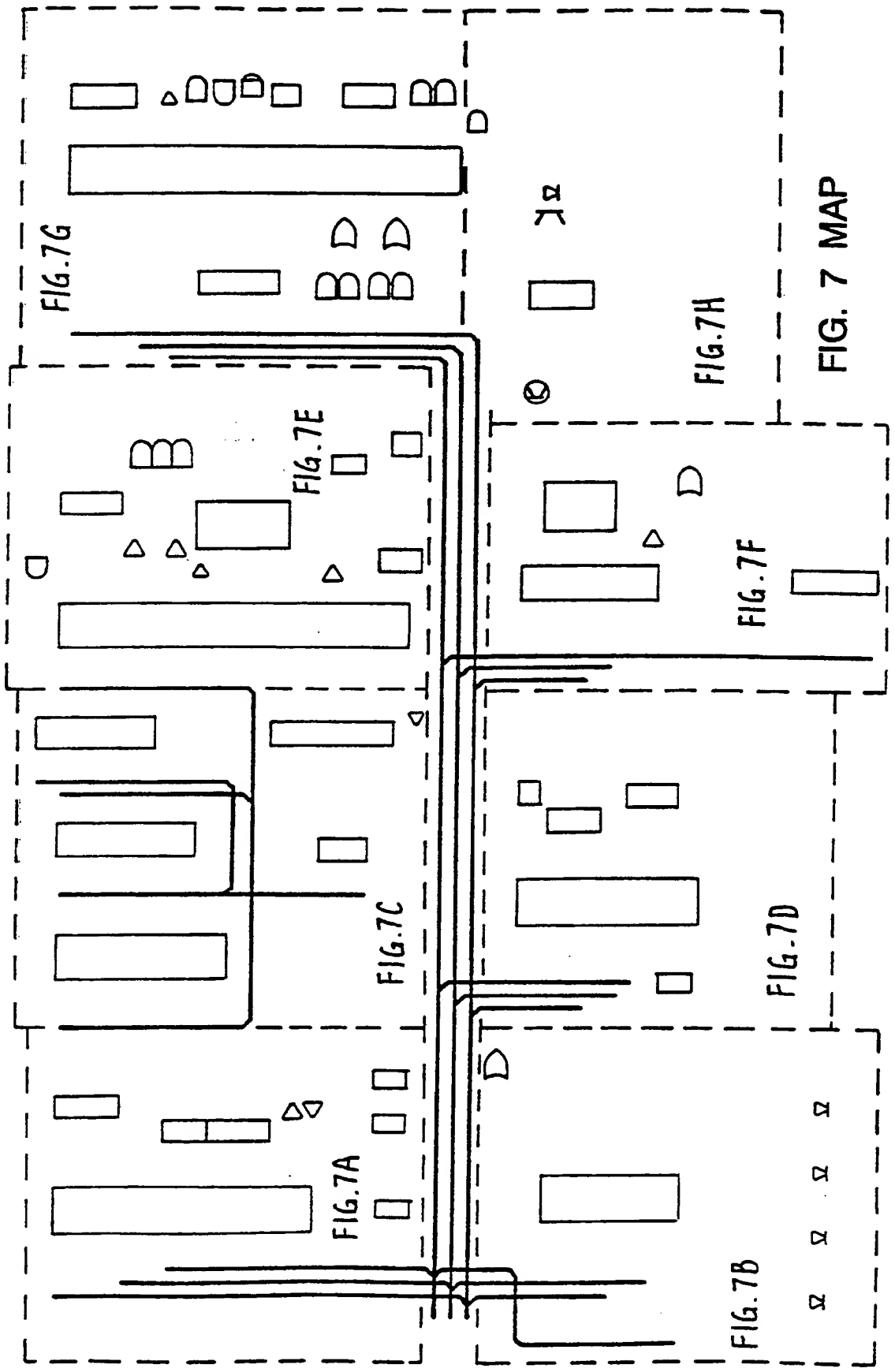


FIG. 7 MAP
FIG.7G
FIG.7H
FIG.7E
FIG.7F
FIG.7C
FIG.7D
FIG.7A
FIG.7B

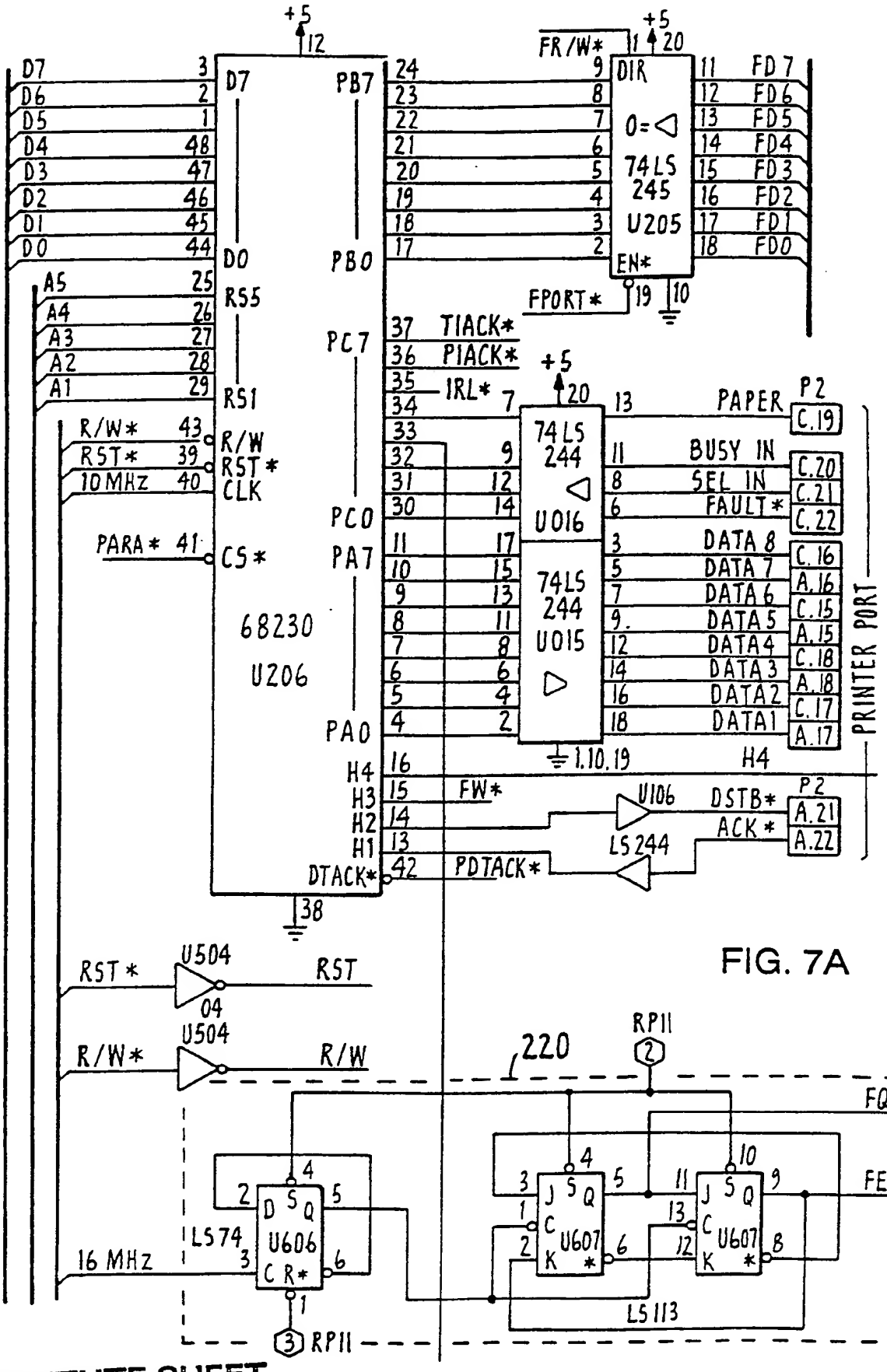


FIG. 7A

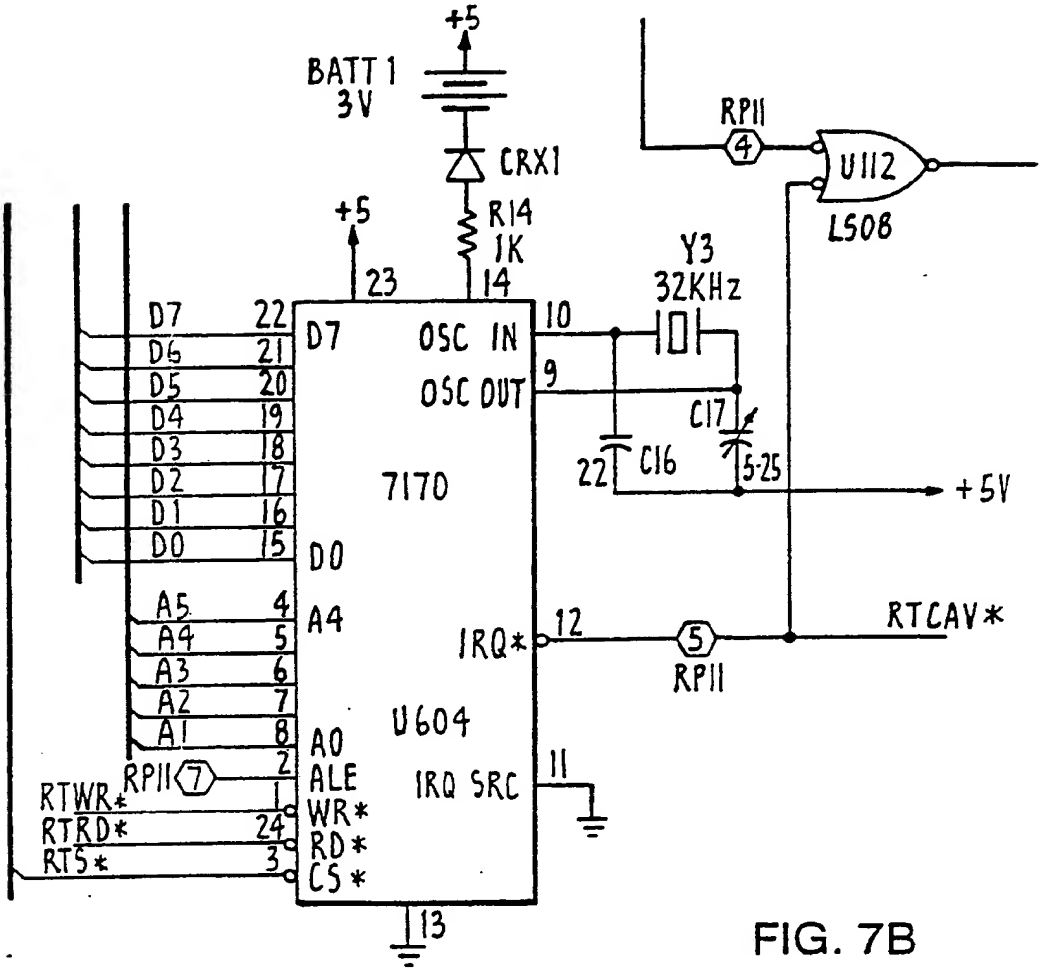
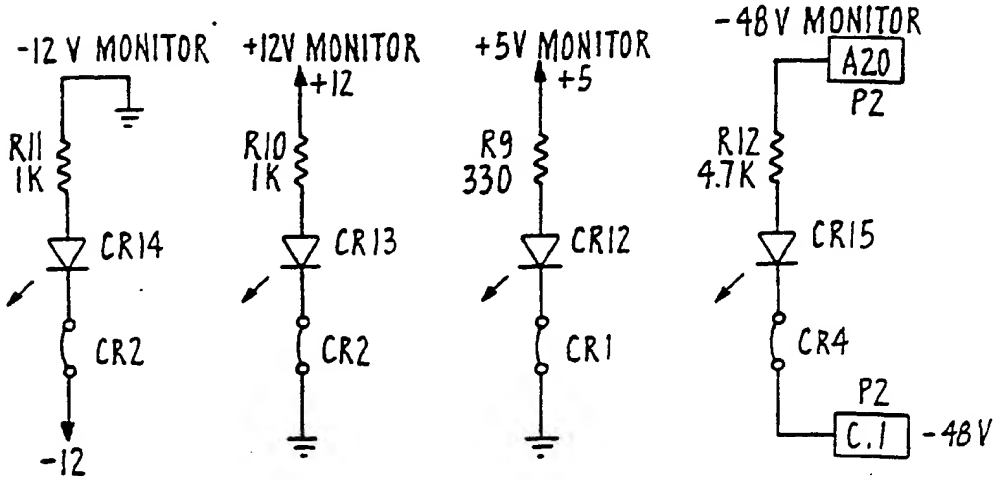

+5
BATT 1
3V
CRX1
R14
1K
RP11
4
U112
LS08
Y3
32KHz
+5
23
14
10
9
C17
22
C16
5-25
+5V
D7
D6
D5
D4
D3
D2
D1
D0
22
21
20
19
18
17
16
15
D7
OSC IN
OSC OUT
7170
D0
A5
A4
A3
A2
A1
A0
4
5
6
7
8
2
A4
A0
ALE
WR*
RD*
CS*
IRQ*
12
5
RP11
RTCAV*
U604
IRQ SRC
11
RP11
7
RTWR*
RTRD*
RTCS*
1
24
3
13

FIG. 7B


-12 V MONITOR
+12V MONITOR
+5V MONITOR
-48V MONITOR
+12
+5
R11
1K
R10
1K
R9
330
R12
4.7K
A20
P2
CR14
CR13
CR12
CR15
CR2
CR2
CR1
CR4
P2
C.1
-48V
-12

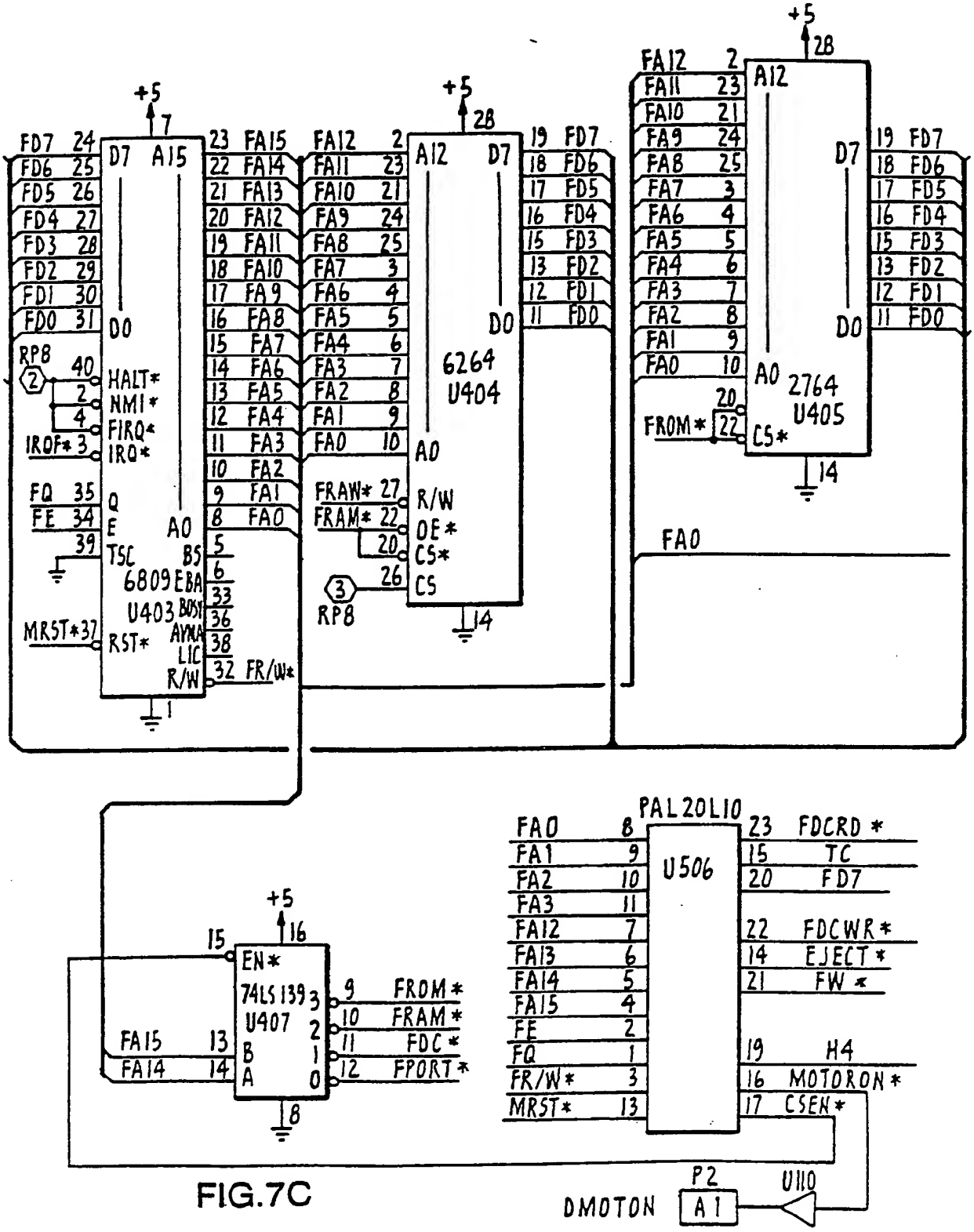

+5
7
FD7 24
FD6 25
FD5 26
FD4 27
FD3 28
FD2 29
FD1 30
FD0 31
D7
A15
D0
RP8
2
40
2
4
HALT*
NMI*
FIRQ*
IRQ*
IROF* 3
FQ 35
FE 34
39
Q
E
TSC
A0
BS
EBA
BOOT
AVMA
LIC
R/W
6809
U403
MRST* 37
RST*
23 FA15
22 FA14
21 FA13
20 FA12
19 FA11
18 FA10
17 FA9
16 FA8
15 FA7
14 FA6
13 FA5
12 FA4
11 FA3
10 FA2
9 FA1
8 FA0
5
6
33
36
38
32 FR/W*
1
+5
28
FA12 2
FA11 23
FA10 21
FA9 24
FA8 25
FA7 3
FA6 4
FA5 5
FA4 6
FA3 7
FA2 8
FA1 9
FA0 10
A12
D7
D0
A0
6264
U404
FRAW* 27
FRAM* 22
20
26
R/W
OE*
CS*
CS
3
RP8
14
19 FD7
18 FD6
17 FD5
16 FD4
15 FD3
13 FD2
12 FD1
11 FD0
+5
28
FA12 2
FA11 23
FA10 21
FA9 24
FA8 25
FA7 3
FA6 4
FA5 5
FA4 6
FA3 7
FA2 8
FA1 9
FA0 10
A12
D7
D0
A0
2764
U405
FROM* 20
22
CS*
14
19 FD7
18 FD6
17 FD5
16 FD4
15 FD3
13 FD2
12 FD1
11 FD0
FA0
+5
16
15
EN*
74LS139
U407
B
A
3
2
1
0
8
FA15 13
FA14 14
9 FROM*
10 FRAM*
11 FDC*
12 FPORT*
PAL20L10
U506
FA0 8
FA1 9
FA2 10
FA3 11
FA12 7
FA13 6
FA14 5
FA15 4
FE 2
FQ 1
FR/W* 3
MRST* 13
23 FDCRD*
15 TC
20 FD7
22 FDCWR*
14 EJECT*
21 FW*
19 H4
16 MOTORON*
17 CSEN*
P2
A1
U110
DMOTON

FIG.7C

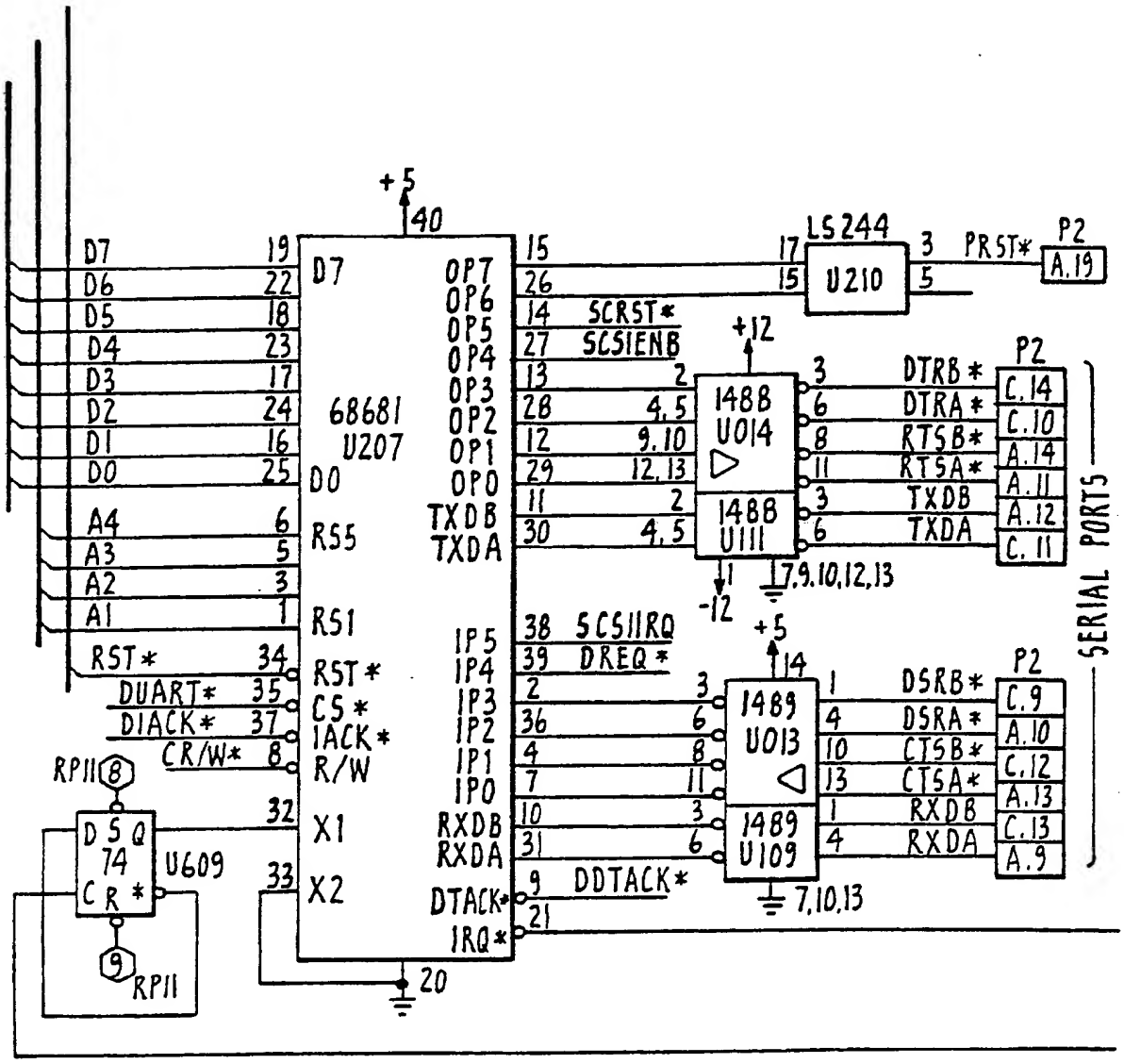

+5
40
D7
D6
D5
D4
D3
D2
D1
D0
A4
A3
A2
A1
RST*
DUART*
DIACK*
CR/W*
RP11
8
D S Q
74
C R *
U609
9
RP11
68681
U207
D7
D0
RS5
RS1
RST*
CS*
IACK*
R/W
X1
X2
20
OP7
OP6
OP5
OP4
OP3
OP2
OP1
OP0
TXDB
TXDA
IP5
IP4
IP3
IP2
IP1
IP0
RXDB
RXDA
DTACK*
IRQ*
SCRST*
SCSIENB
5CSIIRQ
DREQ*
DDTACK*
LS244
U210
PRST*
P2
A.19
+12
1488
U014
1488
U111
-12
7,9,10,12,13
DTRB*
DTRA*
RTSB*
RTSA*
TXDB
TXDA
C.14
C.10
A.14
A.11
A.12
C.11
SERIAL PORTS
+5
14
1489
U013
1489
U109
7,10,13
DSRB*
DSRA*
CTSB*
CTSA*
RXDB
RXDA
C.9
A.10
C.12
A.13
C.13
A.9
+12
P1
C31
P1
A31
-12
+5
P2
B1
B13
B32
P2
B2
B12
B22
B31
12 9 5 2 12 9 5 2 1489 PINS
470
C4 C5 C7 C6 C10 C9 C8 C11
+5V
+12V
C1
C3
-12V
C2

FIG. 7D

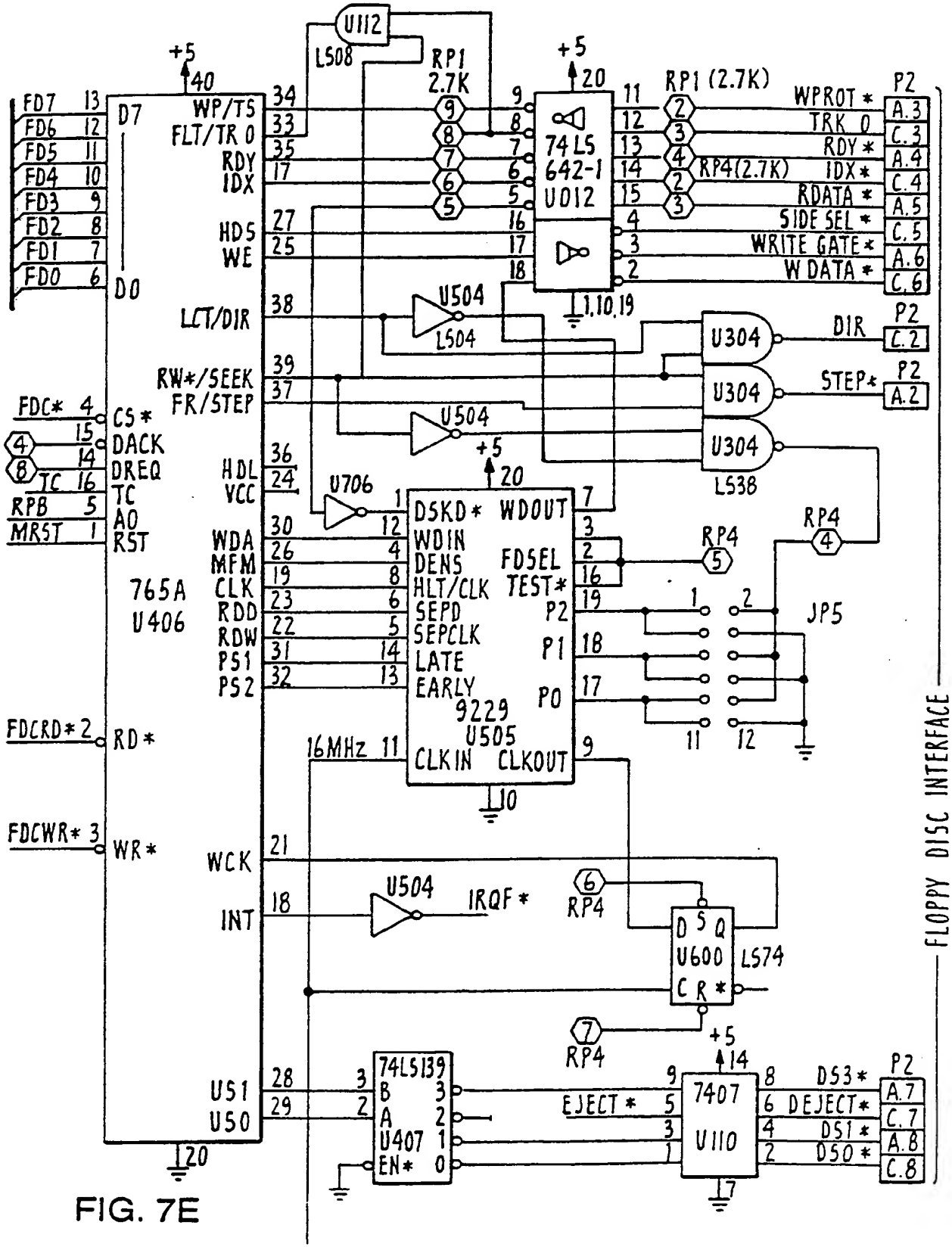


FIG. 7E

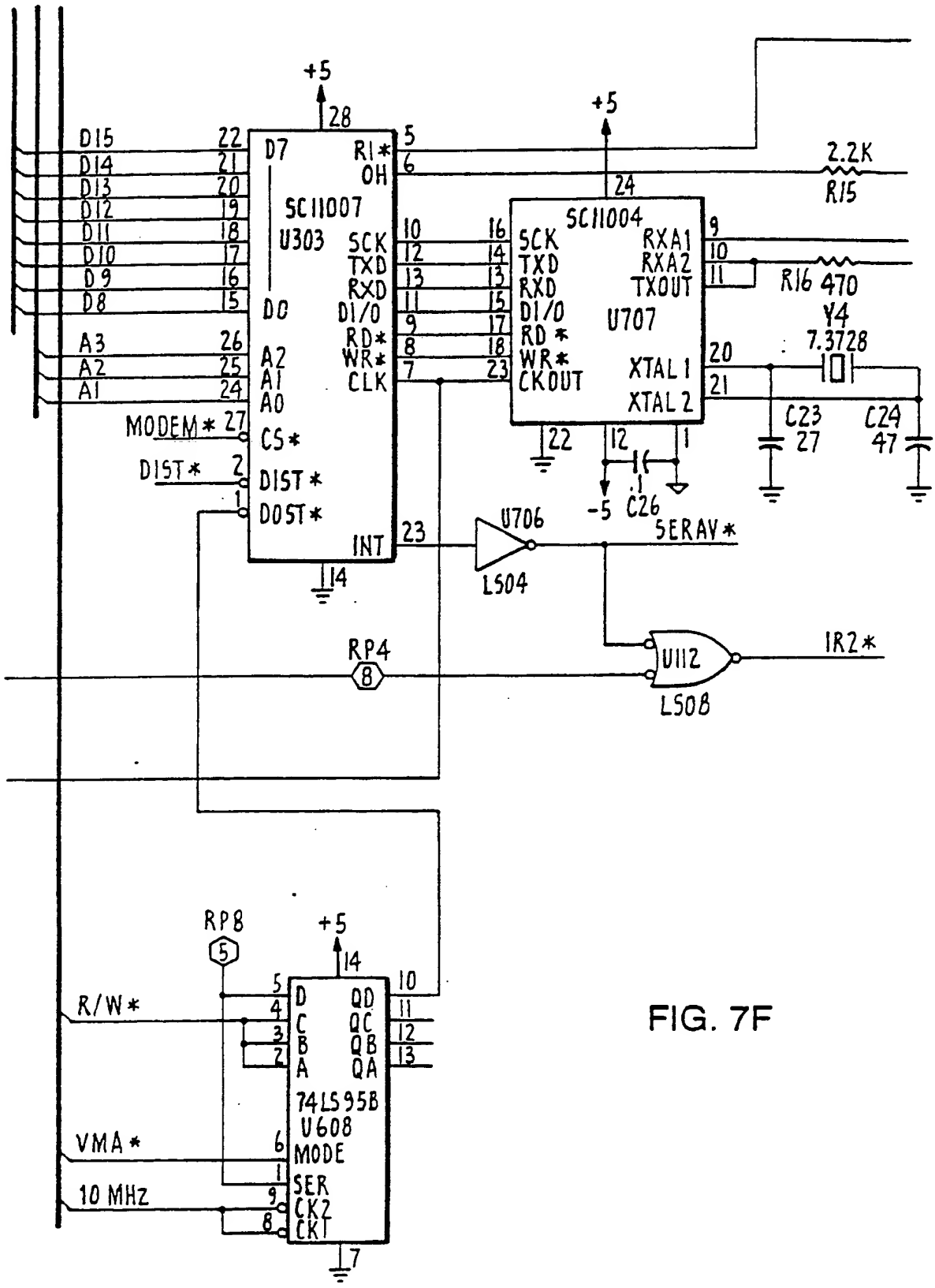

+5
28
D15
D14
D13
D12
D11
D10
D9
D8
A3
A2
A1
MODEM*
DIST*
D7
DO
A2
A1
A0
CS*
DIST*
DOST*
SC11007
U303
SCK
TXD
RXD
DI/O
RD*
WR*
CLK
INT
RI*
OH
14
U706
LS04
SERAV*
RP4
8
U112
LS08
IR2*
+5
24
SC11004
SCK
TXD
RXD
DI/O
RD*
WR*
CKOUT
RXA1
RXA2
TXOUT
XTAL 1
XTAL 2
U707
2.2K
R15
R16 470
Y4
7.3728
C23
27
C24
47
C26
.1
-5
RP8
5
74LS95B
U608
MODE
SER
CK2
CK1
R/W*
VMA*
10 MHz
D
C
B
A
QD
QC
QB
QA

FIG. 7F

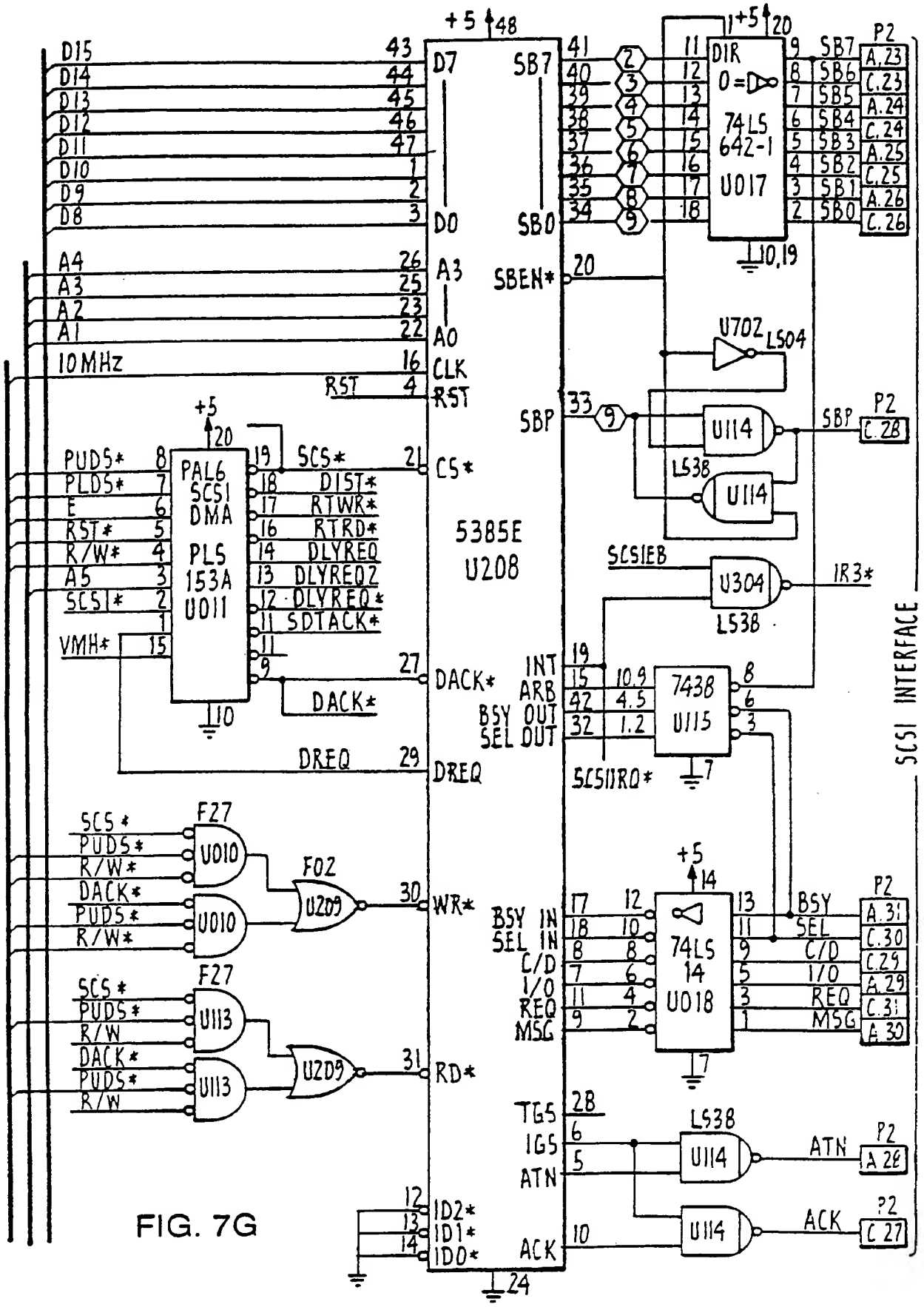

+5
48
D15
D14
D13
D12
D11
D10
D9
D8
43
44
45
46
47
1
2
3
D7
D0
A4
A3
A2
A1
26
25
23
22
A3
A0
10MHz
16
CLK
RST
4
RST
+5
20
PUDS*
PLDS*
E
RST*
R/W*
A5
SCS1*
VMH*
8
7
6
5
4
3
2
1
15
PAL6
SCSI
DMA
PLS
153A
U011
19
18
17
16
14
13
12
11
9
SCS*
DIST*
RTWR*
RTRD*
DLYREQ
DLYREQ2
DLYREQ*
SDTACK*
21
27
C5*
DACK*
DACK*
10
DREQ
29
DREQ
5385E
U208
SB7
SB0
41
40
39
38
37
36
35
34
SBEN*
20
SBP
33
9
SCSIEB
19
INT
ARB
BSY OUT
SEL OUT
15
42
32
10.9
4.5
1.2
7438
U115
8
6
3
7
SCSIIRQ*
IR3*
U304
L538
U114
L538
U114
SBP
P2
C.28
U70Z
LS04
DIR
0=
74LS
642-1
U017
11
12
13
14
15
16
17
18
10,19
9
8
7
6
5
4
3
2
SB7
SB6
SB5
SB4
SB3
SB2
SB1
SB0
P2
A.23
C.23
A.24
C.24
A.25
C.25
A.26
C.26
SCSI INTERFACE
F27
U010
U010
F02
U209
30
WR*
SCS*
PUDS*
R/W*
DACK*
PUDS*
R/W*
F27
U113
U113
U209
31
RD*
SCS*
PUDS*
R/W
DACK*
PUDS*
R/W
+5
14
BSY IN
SEL IN
C/D
I/O
REQ
MSG
17
18
8
7
11
9
12
10
8
6
4
2
74LS
14
U018
13
11
9
5
3
1
BSY
SEL
C/D
I/O
REQ
MSG
P2
A.31
C.30
C.29
A.29
C.31
A.30
7
TGS
IGS
ATN
28
6
5
L538
U114
ATN
P2
A 28
12
13
14
ID2*
ID1*
ID0*
ACK
10
U114
ACK
P2
C 27
24

FIG. 7G

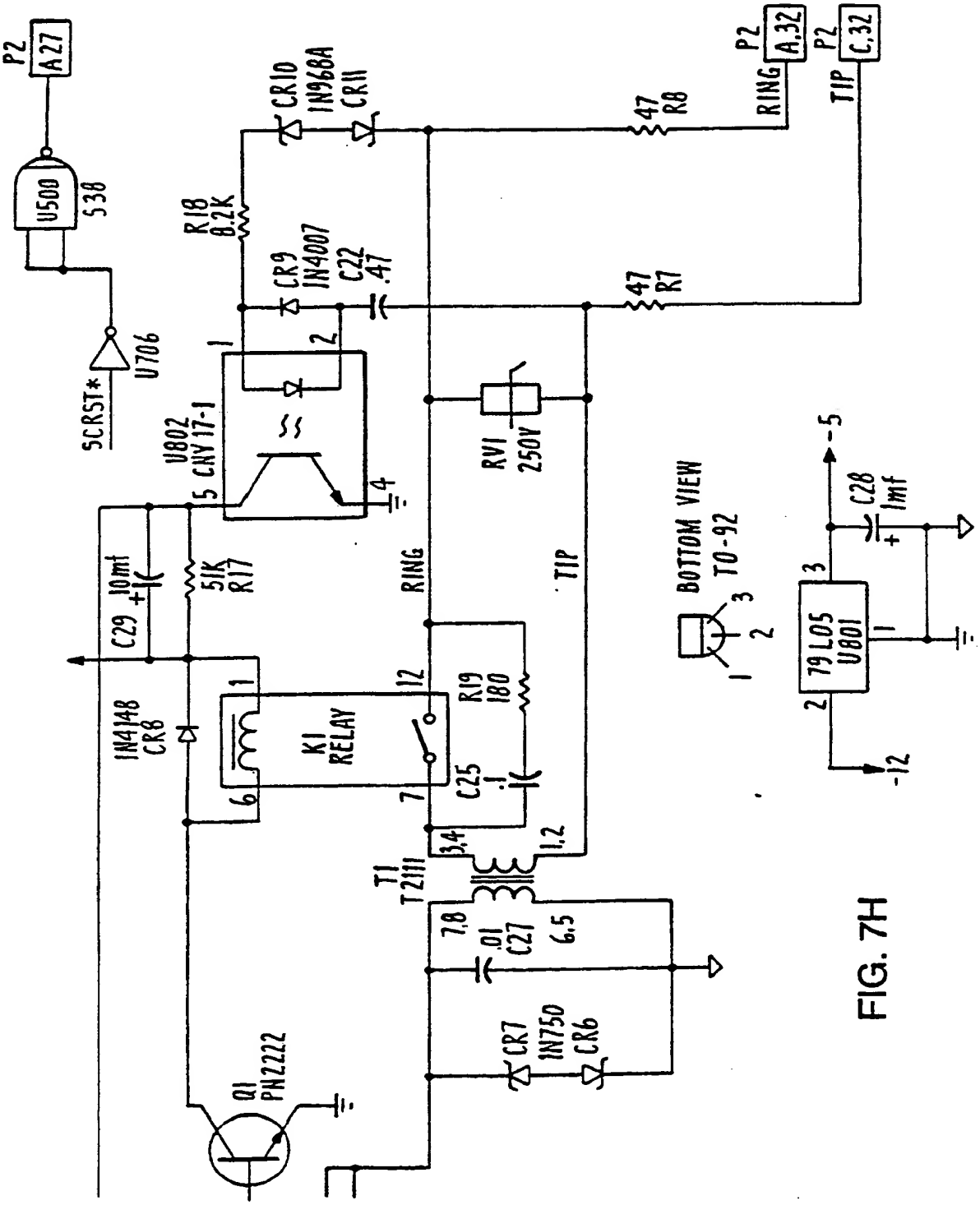

P2 A27
U500 S38
U706
SCRST*
U802 5 CNY17-1
R18 8.2K
CR10 IN968A CR11
CR9 IN4007
C22 .47
47 R8
47 R7
P2 A,32 RING
P2 C,32 TIP
RV1 250V
C29 10mf
51K R17
IN4148 CR8
K1 RELAY
R19 180
C25 .1
RING
TIP
T1 T2111
7,8
.01 C27
6,5
CR7 IN750 CR6
Q1 PN2222
BOTTOM VIEW TO-92
79L05 U801
C28 1mf
-5
-12

FIG. 7H

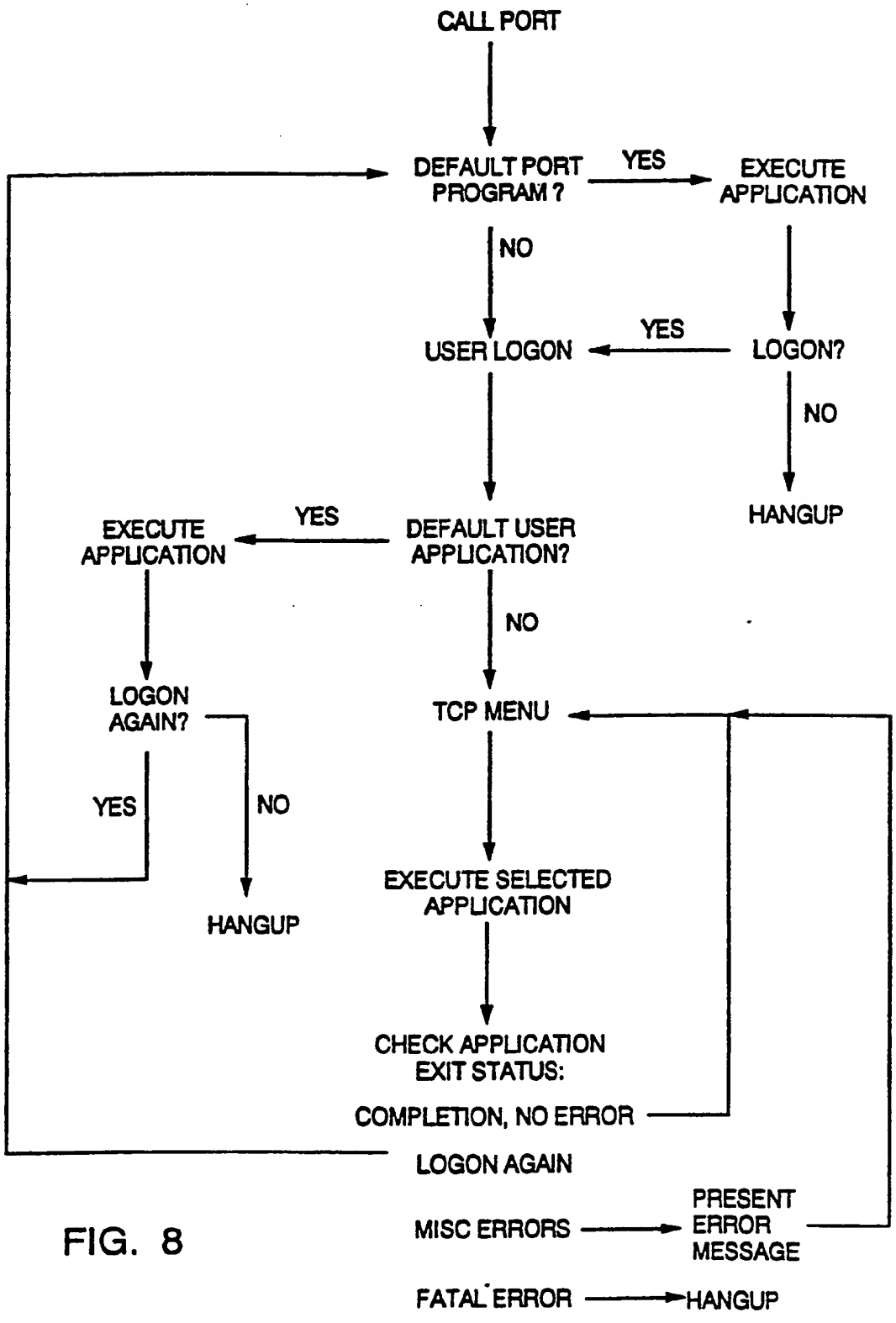
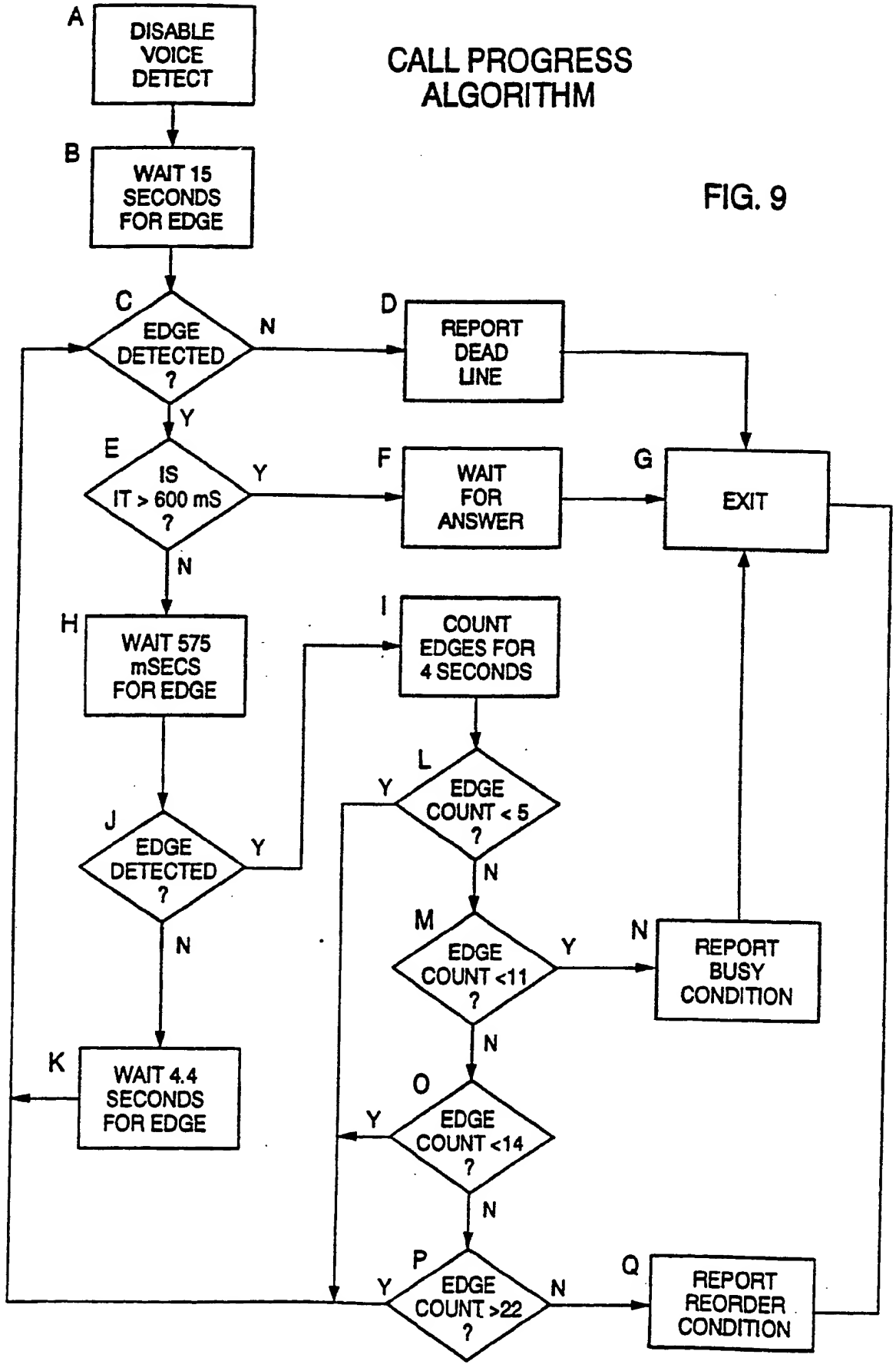

CALL PORT
DEFAULT PORT PROGRAM ?
YES
EXECUTE APPLICATION
NO
USER LOGON
YES
LOGON?
NO
HANGUP
EXECUTE APPLICATION
YES
DEFAULT USER APPLICATION?
NO
LOGON AGAIN?
YES
NO
HANGUP
TCP MENU
EXECUTE SELECTED APPLICATION
CHECK APPLICATION EXIT STATUS:
COMPLETION, NO ERROR
LOGON AGAIN
MISC ERRORS
PRESENT ERROR MESSAGE
FATAL ERROR
HANGUP

FIG. 8

CALL PROGRESS ALGORITHM

FIG. 9



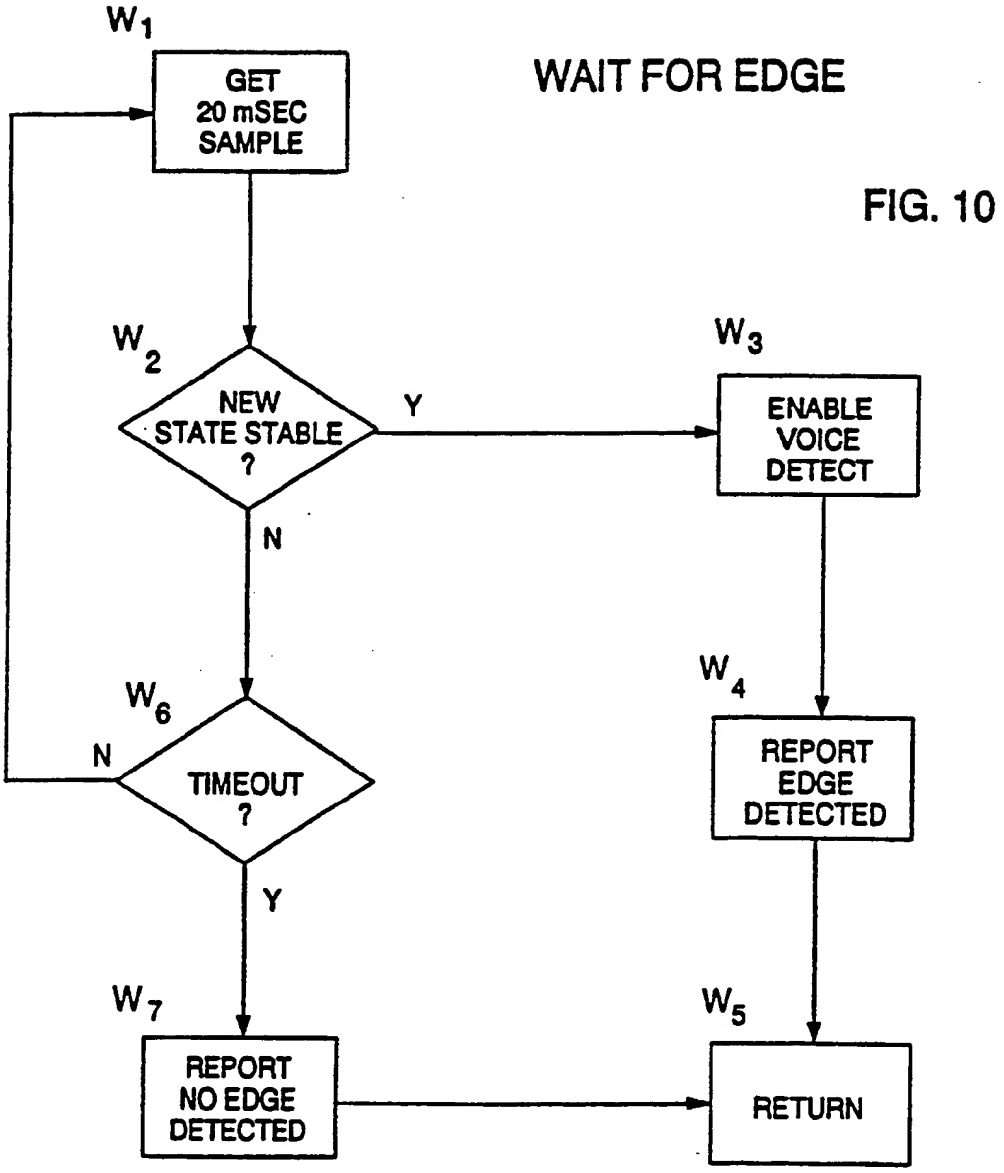

W1
GET 20 mSEC SAMPLE
WAIT FOR EDGE
W2
NEW STATE STABLE ?
Y
W3
ENABLE VOICE DETECT
N
W4
REPORT EDGE DETECTED
W6
N
TIMEOUT ?
Y
W7
REPORT NO EDGE DETECTED
W5
RETURN

FIG. 10

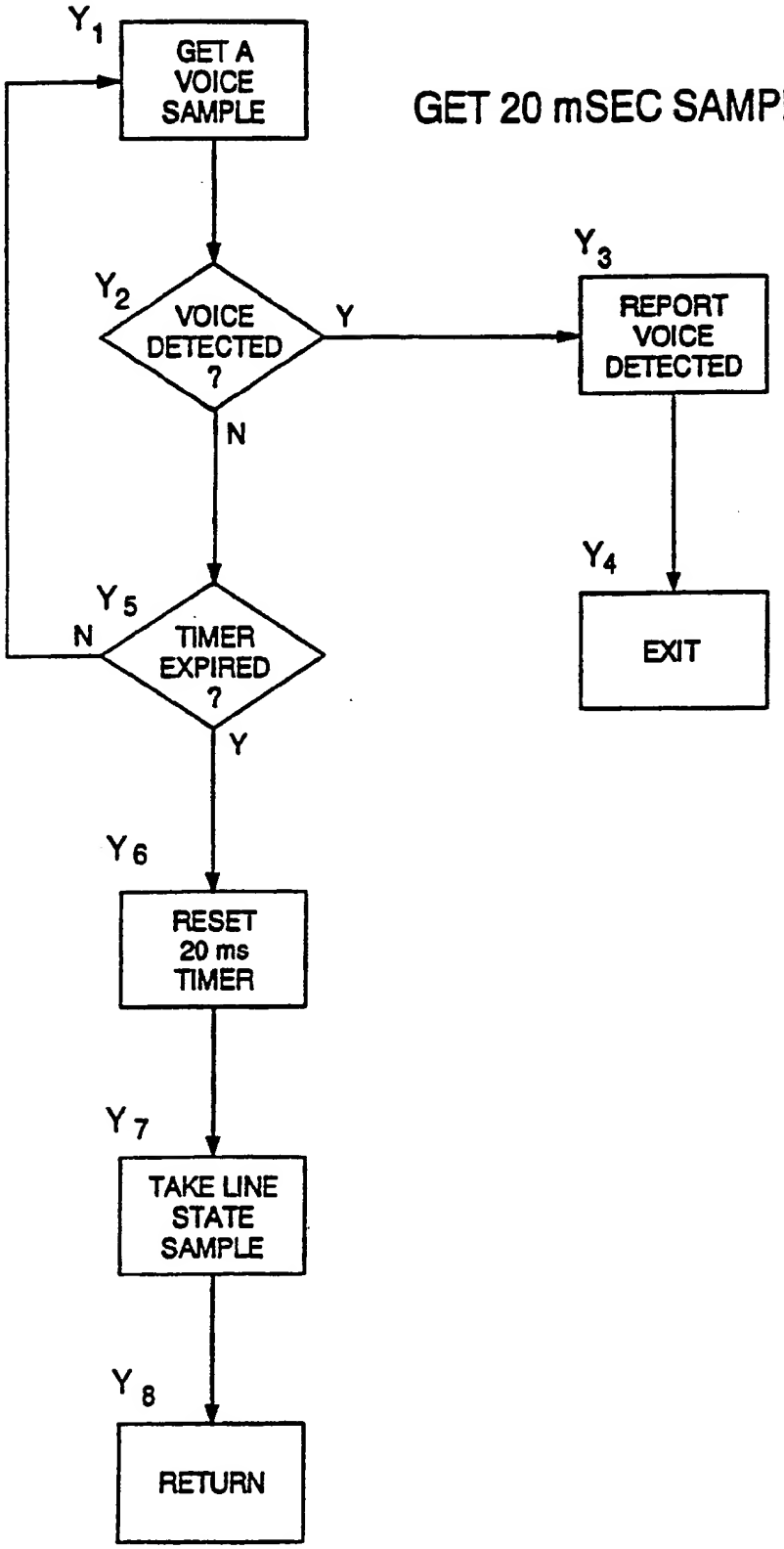
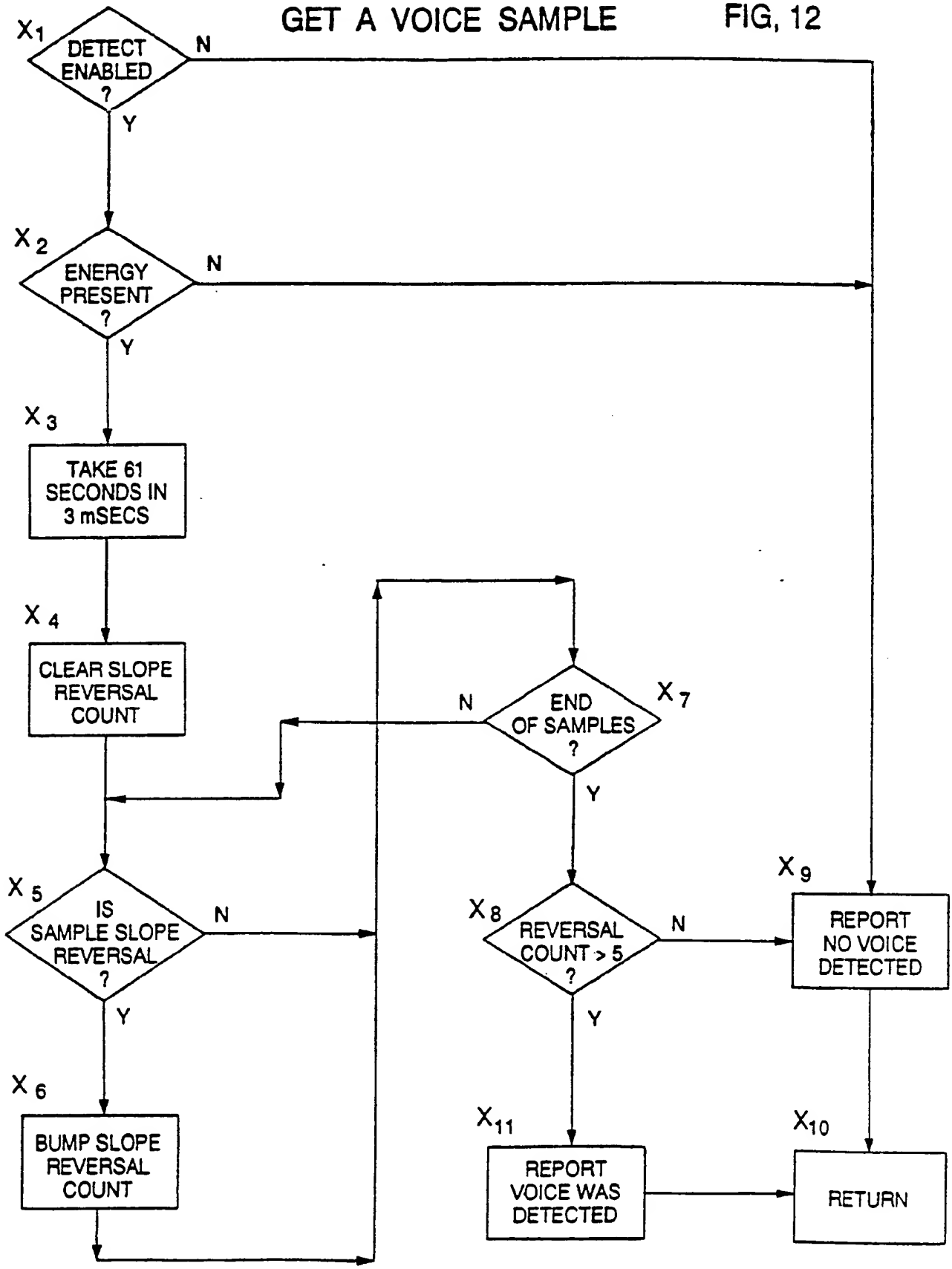
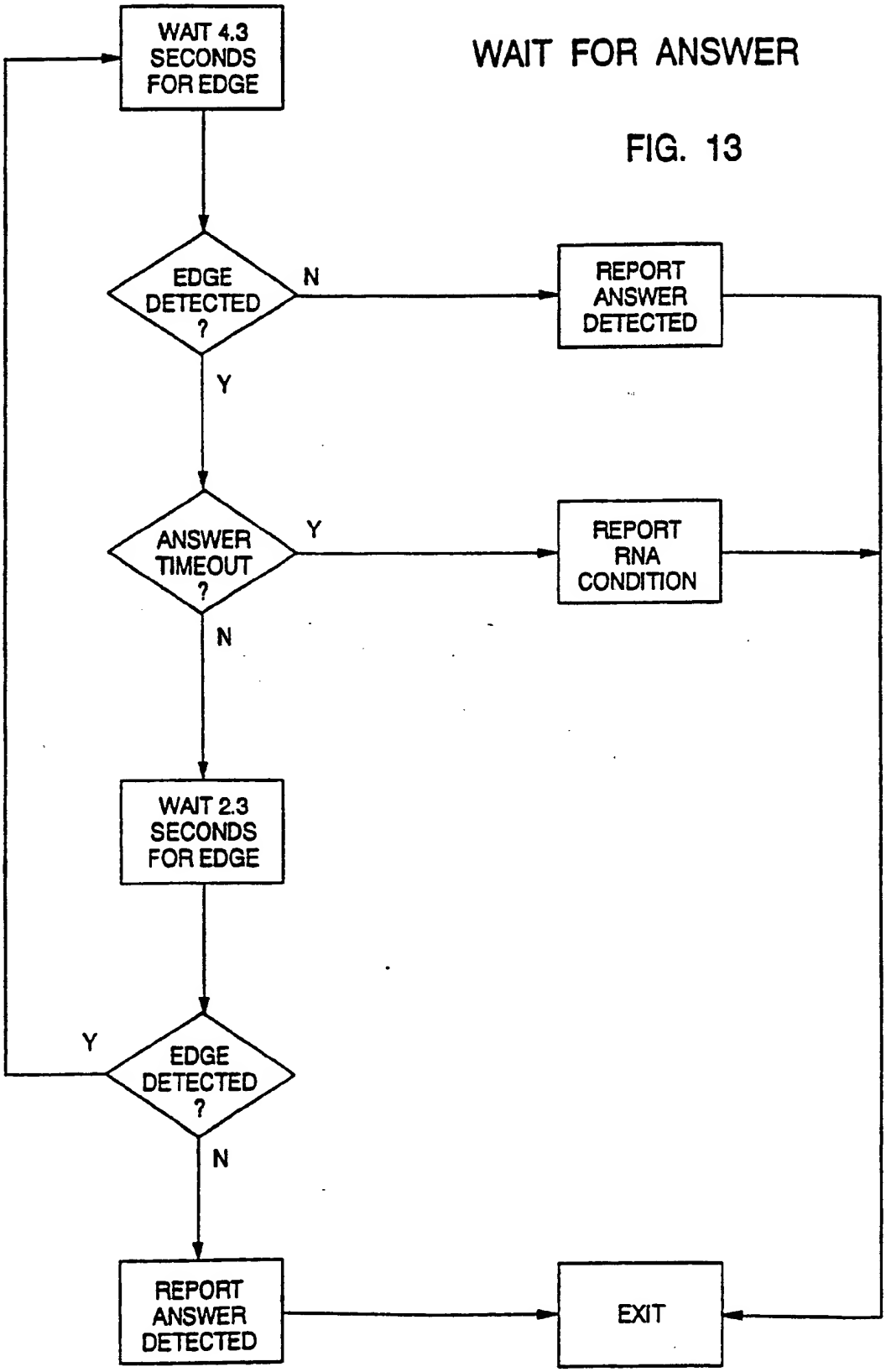

Y1
GET A VOICE SAMPLE
GET 20 mSEC SAMPLE
Y2
VOICE DETECTED ?
Y
Y3
REPORT VOICE DETECTED
N
Y4
EXIT
Y5
N
TIMER EXPIRED ?
Y
Y6
RESET 20 ms TIMER
Y7
TAKE LINE STATE SAMPLE
Y8
RETURN

FIG. 11

GET A VOICE SAMPLE FIG, 12





WAIT FOR ANSWER

FIG. 13

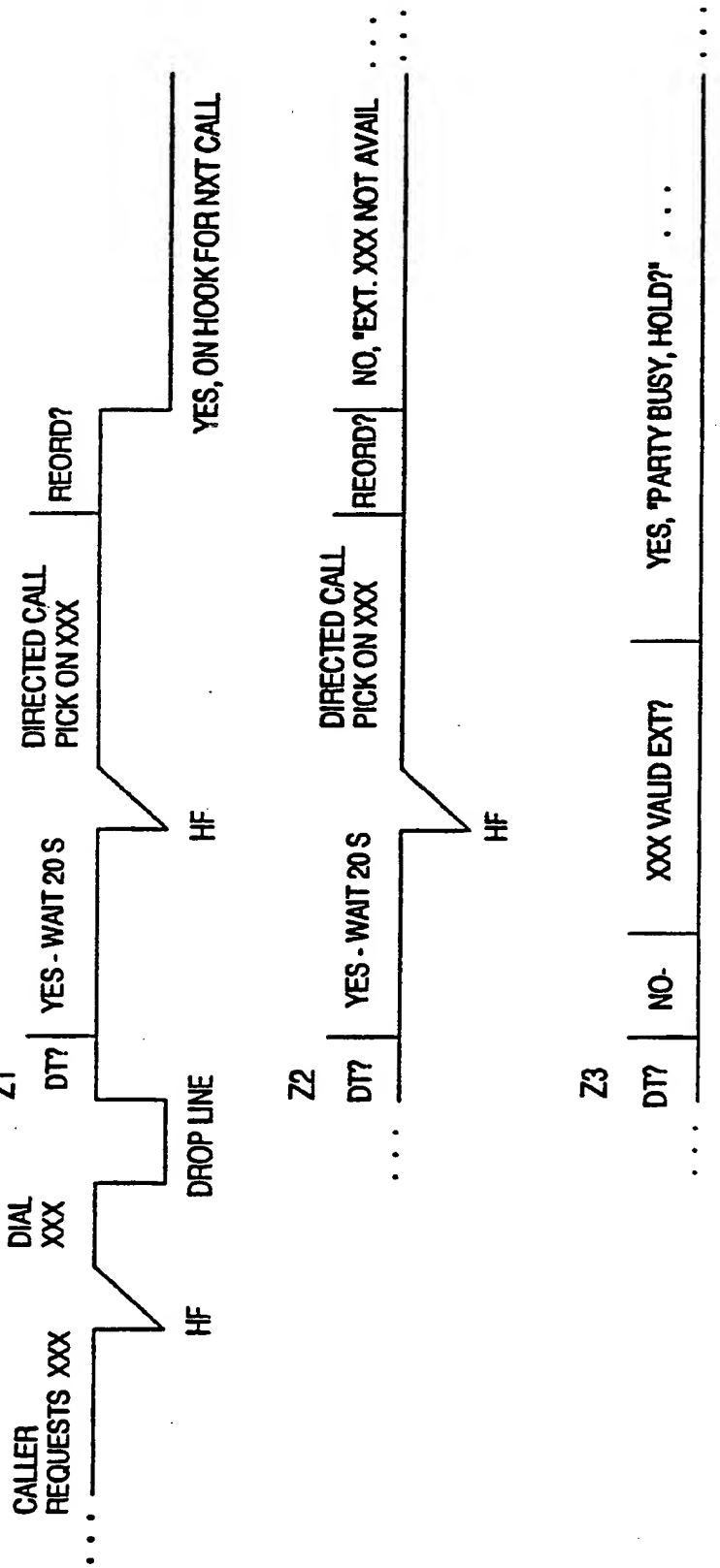

CALLER REQUESTS XXX
HF
DIAL XXX
DROP LINE
Z1
DT?
YES - WAIT 20 S
HF
DIRECTED CALL PICK ON XXX
REORD?
YES, ON HOOK FOR NXT CALL
Z2
DT?
YES - WAIT 20 S
HF
DIRECTED CALL PICK ON XXX
REORD?
NO, "EXT. XXX NOT AVAIL
Z3
DT?
NO-
XXX VALID EXT?
YES, "PARTY BUSY, HOLD?"

FIG. 14

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US89/02250

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC(4): G06F 1/00; H04M 3/00

U.S. CL. 364/200; 364/900, 379/88; 379/96

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
US	364/200, 364/900, 379/88, 379/96

Documentation Searched other than Minimum Documentation to the Extent that such Documents are included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X,P	US, A, 4,755,932 (DIEDRICH) 5 July 1988.	1-46
X	US, A, 4,549,047 (BRIAN ET AL.) 22 October 1985.	1-46
X	US, A, 4,375,083 (MAXEMCHUK) 22 February 1983.	1-46
X	US, A, 4,554,418 (TOY) 19 November 1985.	1-46
X	US, A, 4,071,888 (OWENS) 31 January 1978.	1-46

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
10 October 1989	07 NOV 1989
International Searching Authority	**Signature of Authorized Officer**
ISA/US	JOHN G. MILLS

Form PCT/ISA/210 (second sheet) (Rev.11-87)